EuroSPI´99 Conference

European Software Process Improvement

Theme 99 :

" Learn from the Past - Experience the Future "

With the participation of :

B-K Medical, Centraal Beheer, Centre de Recherche Public
Henri Tudor, COPENHAGEN Business School, Cortis Lentini, CSELT, Dakosy, Daxware, DCU,
DRA,EDVGesmbH, Eliop, Ericsson Telecommunicazioni, European
Commission, Event AS, GMV, Hünsberg AG, Hyperwave,
Intecs Sistemi S.p.A, Intracom, Kapital IT, ISG, IVM, Konsberg Ericsson, NASA, NOKIA,
Provida, ReisRobotics, Riga InformationTechnology Institute,Sainsel,

SALFORD City Council, SEKAS, SOUTHAMPTON Institute,
Sztaki, Telenor, The Trentino Federation of Cooperatives,
Universities of
DERBY, LIMERICK, MANCHESTER, MARIBOR,
North LONDON, OSLO, SALFORD,STOCKHOLM,
TAMPERE, WARWICK

25-27 October 1999

Pori School of Technology & Economics
Pohjoisranta 11, Pori, FINLAND

organised by
DELTA *(http://www.delta.dk)*, IVF *(http://www.ivf.se)*, ISCN *(http://www.iscn.ie)*
SINTEF *(http://www.sintef.no)*, STTF *(http://www.sttf.fi)*

*We would like to thank the European Commission to support
us in the integration of PIEs into this conference*

Conference Web Site : http://www.bigfoot.com/~EuroSPI

Contents:

Session 1 : SPI and Strategies

A Total Improvement Strategy as a Basis for Software Process Improvement 1.2
Key Success Factors for Business Based Improvement 1.16
SPI – Why isn't it more used? 1.34
Software Measurement Frameworks to Assess the Value of Process Improvement 1.50

Session 2 : SPI and Testing 1

The QUEST for Quality Test Resources 2.2
Software Inspection Techniques in SMEs 2.18
Productivity Improvement via Software Testing 2.34
Practical Measurements for Reengineering the Software Testing Process 2.48

Session 3 : SPI and Re-Use

Finding a Practical Approach to Organised Reuse 3.2
SEPIOR - Practical Experiences with Reusable CAM Components 3.26

Session 4 : SPI and Requirements Management

The user requirements elicitation and specification process 4.2
Achieving Customer Satisfaction through Requirements Understanding 4.15

Session 5 : SPI and Establishment of Models/Processes I

Process Description and Training: The Two Sides of the SPI? 5.2
The impact of a new software development methodology and how to afford the changing resistance 5.13
Software process improvement using CASE: lessons from the front-line 5.26

Session 6 : SPI and Virtual Team and QA Systems

Experience with PI in Distributed Virtual Work Environments 6.2
Results from the ESSI process improvement experiment -Virtual Team 6.20
Automating the Use of a Quality System 6.34

Session 7 : SPI and Assessments / Evaluations

SPI Risk Assessment 7.2
Rethinking the Concept of Software Process Assessment 7.11
Philosophies and Approaches to Software Process Improvement 7.24
The General Effect of an Integrated Software Product Evaluation 7.39

Session 8 : SPI Surveys

Software Process Improvement Network in the Satakunta Region- SataSPIN 8.2
Mid-term results of the SPIRAL Network Development 8.12
Software Engineering in the UK – A Brief Report 8.29
An Overview of the SPI Activities in Estonia 8.41

Session 9 : SPI and Establishment of Processes/Models II

Configuration Management Deployment & Practice Experiment 9.2
Quality Assurance for NC-Software with the support of Configuration Management 9.12

Software Process Improvements in a Very Small Company 9.24
Evolutionary Development Process – A Case Study 9.40

Session 10 : SPI and Measurement I

Applying Gilb's method of inspections into telecommunications software 10.2
Control your projects by improved planning 10.18
An extranet for the improvement of the Outsourcing of Software Maintenance projects using Function Points Analysis 10.29
Software Metrics for Process Improvement Experiments 10.52

Session 11 : SPI and Measurement II

SUPREME – a Statistical Approach to Support Project Estimation and Management 11.2
Introducing professional project management in an SME 11.17
Experience from process improvement in a SME 11.28
Error Trending - Why and How 11.41

Session 12 : SPI and Strategies

People Management and Development Process to motivate, develop and retain the best resources 12.2
A Learning Organisation Approach for Process Improvement in the Service Sector 12.14
Process Improvement: The Societal Iceberg 12.34
IPSSI: A European Methodology on PSP 12.44

Session 13 : SPI and Object Orientation

Improvement of the Quality in the Software Development Process by the Introduction of UML 13.2
Improvement of Extendibility and Modifiability of Embedded Software 13.12

Session 14 : SPI and Testing II

Improvement of development process through enhanced test procedure & change request management 14.2
Improvement of testing process through systematisation for increasing software manufacturing assurance 14.22

Conference Board

Tor Stalhane, Sintef, Norway
Risto Nevalainen, STTF, Finland
Carsten Jorgensen, Jorn Johansen, Delta, Denmark
Yingxu Wang, IVF, Sweden
Richard Messnarz, ISCN, Ireland and Austria

International Organiser

Richard Messnarz, ISCN, Ireland

Local Organiser

Timo Varkoi, Pori School of Technology, Finland

Session 1
SPI and Strategies

Chairman
Timo Varkoi
Pori School of Technology, Pori, Finland

A Total Improvement Strategy as a Basis for Software Process Improvement

Rolf H. Westgaard
Solveig Gustad
***Kongsberg Ericsson Communications** ANS*
P.O. Box 87
N-1375 BILLINGSTAD
Norway

Introduction

Kongsberg Ericsson is working with improvements at several company levels, driven from management down and from the "floor" and up.
This paper describes:

- how software process improvements tie up with strategic improvement plans
- some results of our participation in the Norwegian SPIQ project

We decided to join the SPIQ (Software Process Improvement for better Quality) project in 1997. SPIQ is a company driven research and development project, which is anticipated to last until 2001. 10 to 15 companies and 3 research institutes work together to adapt and try out modern methods for systematic process improvement with focus on software quality.

The reason for our participation is a desire to improve our system for continuous improvement of the development processes - starting with software. We were looking for a system, which could give us objective data as a basis for improvement. Before joining SPIQ our actions to improve the development processes had been mostly based on single cases, without any statistics as a basis for priority of actions.

Our Values as a Guide

We have decided that the following values are of utmost importance for our success:

1. *Competence*

We shall have world class competence, which is decisive for our ability to compete

2. *Customer Orientation*

We shall be better than our competitors regarding customer orientation and nearness to our customers

3. *Quality*
We shall continuously identify areas for improvement and we shall prove that we have the ability to carry out improvements within these areas
4. *Concentration*
We shall maintain focus; i.e. we shall only work within the areas that are given priority by company strategies

The Business Process Model

The way we obtain our results is simplified in the Business Process Model below.

Strategic Plan Structure = Organisation Structure = Business Structure:

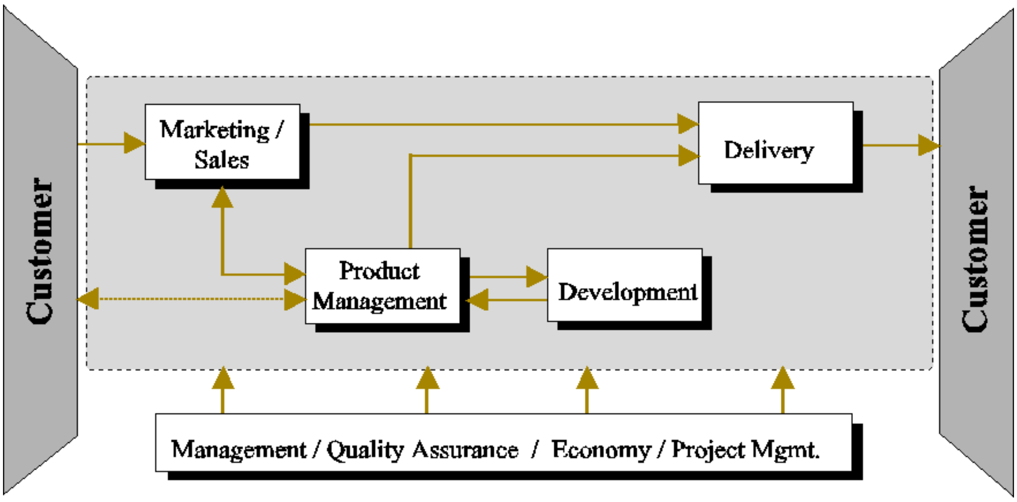


Fig. KEC_RHW 1: The KEC Business Process Model

- Our customers are at the beginning and end of our Business Process Model
- We are organised in a way, which supports our business processes
- Our strategic plan coincides with the organisation structure and the business processes

The Strategic Planning Wheel

The company has established a Strategic Planning Wheel with a one-year cycle. The wheel consists of the following sequences:

- Making or modifying the strategic plan for the coming 4 - 5 years
- Defining next year budget
- Defining goals including plans of action for the next year

The strategic plan part of the wheel includes a Strategic Improvements Plan. This is based on our business processes and the EFQM business excellence model.

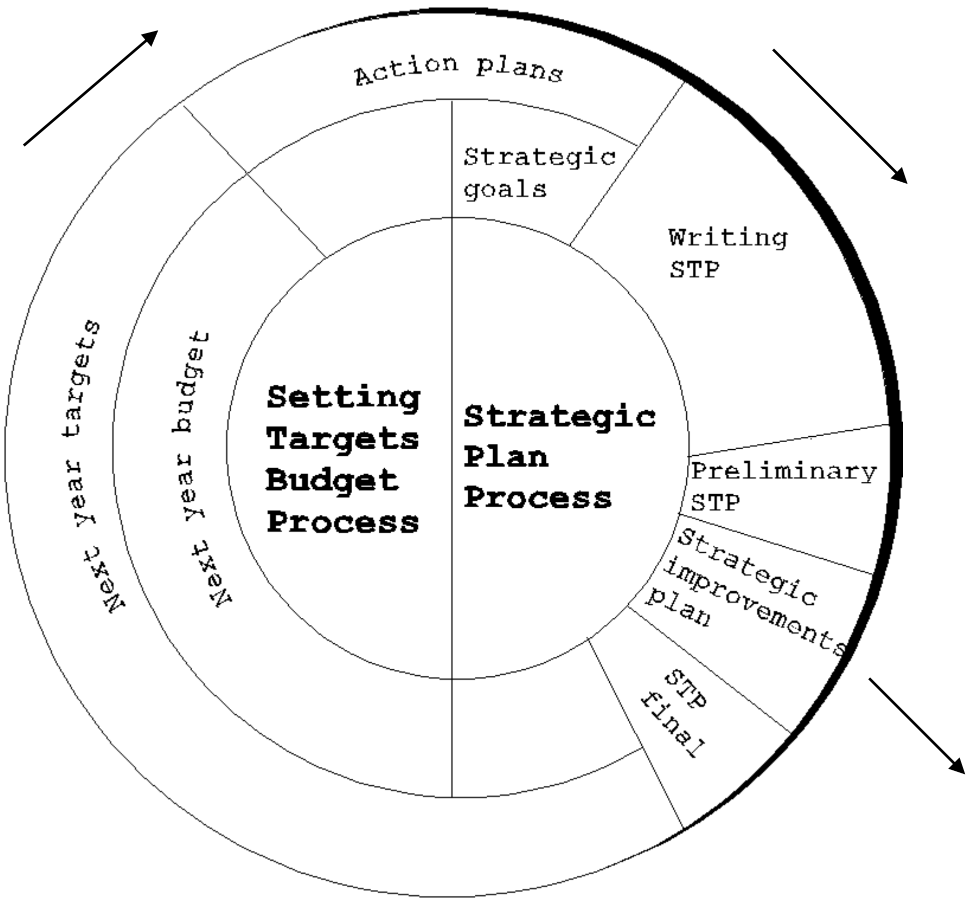


Fig. KEC_RHW 2: The Strategic Planning Wheel

The Strategic Improvements Plan

The Strategic Improvements Plan is based on our business processes and the EFQM business excellence model. Our version of the EFQM business excellence model has added “Opportunities” to it, and we say, *“We are using our market, product and technological OPPORTUNITIES by means of ENABLERS to obtain RESULTS”.* During the Strategic Plan Process we decide the areas within “enablers” that need improving based on long term and short term results. The conclusions are documented in improvement plans starting at company level and broken down into lower organisational levels. The plans are finally the basis for each person’s individual contribution to improvements, agreed during the yearly appraisal interview.

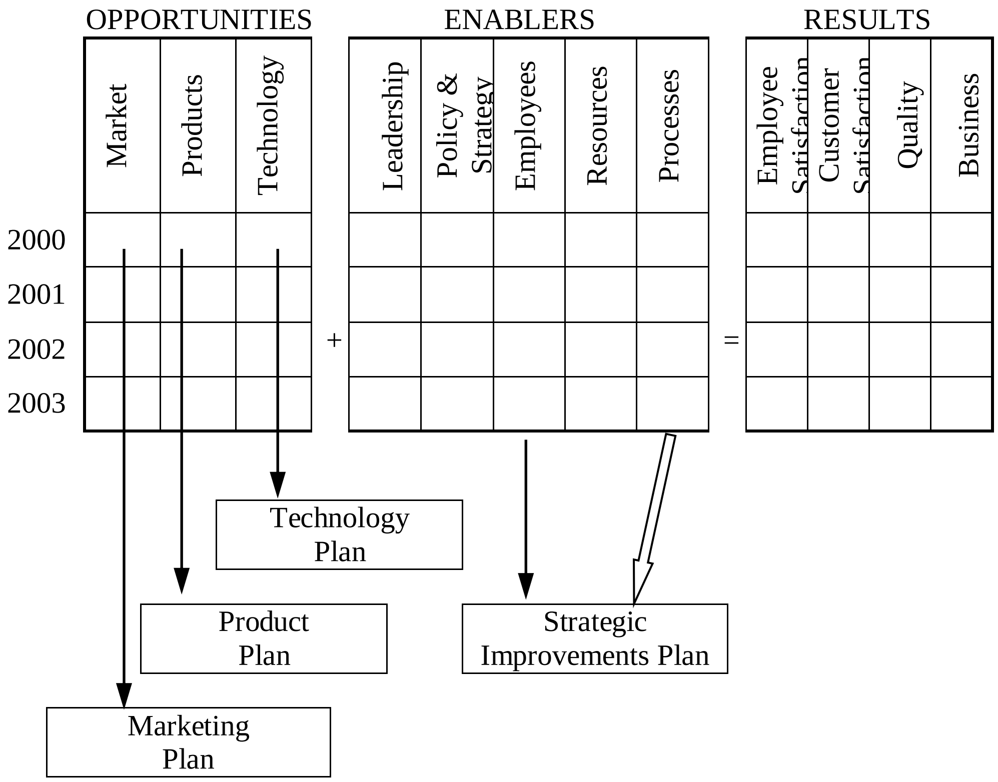


Fig. KEC_RHW 3: Modified EFQM Business Excellence Model

Follow-up of Improvement Plans

The improvement plans at the different levels contain:

- goal definition
- activities to reach the goal
- deadlines
- responsible

and are followed up in regular management meetings:

Opportunities: Market & Product meetings every two weeks
Enablers: Value meetings every month
Results: Result meetings every month

We have all experienced that a well-structured plan and good intentions do not necessarily guarantee results. Therefore, we have defined some critical factors to secure success:

1. The Company President must be the owner of the improvement processes, and he must show the way
2. Make it simple, and make sure the goals are realistic
3. The timing must be right
4. The person who is responsible for reaching the goal should be the one who defines it through a process of involvement and enthusiasm

Process Improvements

The most comprehensive processes in our company are the ones covering product development. Improving efficiency and quality of these processes is therefore a high priority.

The following part of the paper describes our efforts to establish a system for improvement of our development processes, starting with the software process and our participation in SPIQ with focus on software of the MRR project.

Software process improvement based on measurements

This case describes the planning, implementation and results of a software process improvement project. The company has long experience of collecting data, but not to assemble the data and use them to extract statistics and get experience from them.

What have we done?

- GQM-method to decide what to improve, what to measure and how
- Pareto analysis to analyse the data and identify improvement actions

During 9 analyse meetings, 545 error reports have been analysed. About 14 software developers have been involved in addition to the software manager, the quality manager and the software project manager.

The project

The MRR (Multi Role Radio) project is a large electronic project of 1,9 Billion NOK (230 Million EUR). The product is a portable radio for military use. The project started in 1989 and is a co-operation between Kongsberg Ericsson and Thomson CSF Norcom. The pilot project described in this paper involves only the Kongsberg Ericsson part of the MRR project. MRR was delivered in 1996 as a pre-production model with reduced functionality. After that all HW and almost all SW has been redesigned.
Because the MRR project has lasted for almost ten years, there have been some turnover in labour. The development process is an ordinary waterfall model and this model was the starting point for improvements.

The starting scenario

Most of the software was finished in the start of the SPIQ project, only testing remained. That meant that almost all the errors were already done.

To correct all errors, there was established a “system acute” that handles all error reports every day at 12 a.m. From this meeting the error report is decided whether to correct or not and the report will be sent to the responsible person with some action (usually correct a document or code). In this way we know that every error detected in test will be handled. But there is no analysis of the errors to prevent similar errors in the future. Here is where the SPIQ project comes in.

The starting point is our written development process (all procedures, checklists, rules, templates, the tools we use, language, the methods and so on). If we don’t change anything of this, we will not be better in the future. So we had to change something, but what? And when would we know that a change had a positive effect?

Work done

To answer the questions above, we joined the *measurement and experience-data-*group in SPIQ. We started with a GQM workshop for all the software developers. The result of this workshop was a GQM abstraction sheet with one goal, eight questions and 18 metrics. We need to answer the five questions in order to reach the goal, and we need to collect the 18 metrics in order to answer the questions. The abstraction sheet also contains a plan of measurements (what should we measure, how and who etc.).

GQM example

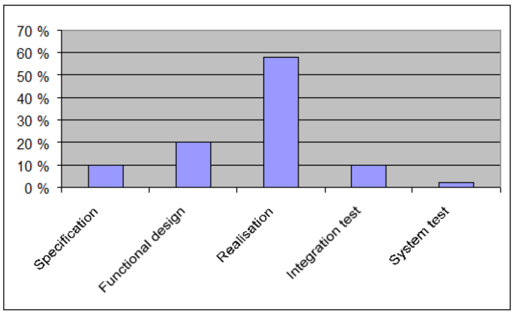
The main goal was to *achieve reduction in the number of serious errors found in integration test and system test by 50%.*

The GQM goal then became to *analyse the development process in order to reduce the number of serious errors introduced before FAT (Factory Acceptance Test) seen from the software group in the environment of the MRR project*.

GQM abstraction sheet:

<table>
<tr><td colspan="2"><b>Analyse the development process in order to reduce the number of serious errors seen from SW group in the MRR project</b></td></tr>
<tr><td><b>Quality Focus</b>

Q1 In which phase was the error introduced?
Q2 Why was the error introduced (the cause)?
Q3 What is the cost distributed over phase/products?
Q4 What is the distribution over error categories?
Q5 What is the error density
– totally?
– per module/product?</td><td><b>Validation Factors</b>

Qa What is the complexity of: project, product and modules?
Qb What is the quality level of the technology/tools?
Qc What is the knowledge level in the project?</td></tr>
<tr><td><b>Baseline Hypothesis</b>
Q1:
 </td><td><b>Impact on Baseline Hypothesis</b>

Qa Increased complexity →
- More errors (increased Q5)
- Errors introduced in earlier phases (Q1 shifts to left)
– Increased error cost (increased Q3)
Qb High quality technology/tools → positive effect on Q1, Q4 and Q5
Qc Sufficient qualifications in the project → positive effect on Q1, Q2 and Q5</td></tr>
</table>

Measurement plan:

Question	**Q1 In which phase was the error introduced?**
Metric	M1: Phase Introduced
Definition	
Presentation and analysis	Bar chart (one bar per phase)
Baseline hypothesis	See GQM abstraction sheet

Question	**Q2 Why was the error introduced (the cause)?**
Metric	M2: Error cause
Definition	
Presentation and analysis	Bar chart (one bar per error cause)
Baseline hypothesis	

Metrics:

Metric	**M1 Phase introduced**
Definition	The phase, where the oldest document that has to be changed, was created
Procedure for collecting	Each error shall be classified in one of the following phases: • Phase 2 (system specification) • Phase 3 (functional design) • Phase 4 (realisation) • Phase 5 (functional verification) • Phase 6 (system verification)
When	Each time an error is corrected
Expected value	
Responsible	The person who is responsible for correcting the error

Metric	**M2 The cause of the error/change**
Definition	
Procedure for collecting	Each error/change shall be classified in one of the following causes: 1. A wish for a change or improvement 2. Lack of system knowledge 3. Carelessness/heavy time schedule 4. Lack of or poor source document 5. Other causes
When	Each time an error is corrected
Expected value	
Responsible	The person who is responsible for correcting the error

Results from two of totally five focus questions

About every six weeks, we analyse the error reports and define actions. The actions are mostly of three kinds:

- adjust/change the improvement process itself
- change the development process (i.e. procedures, review rules, checklists)

- collect more information
- improve knowledge

The results after 9 analysis meetings and 545 errors are shown below.

In which phase are the errors introduced?

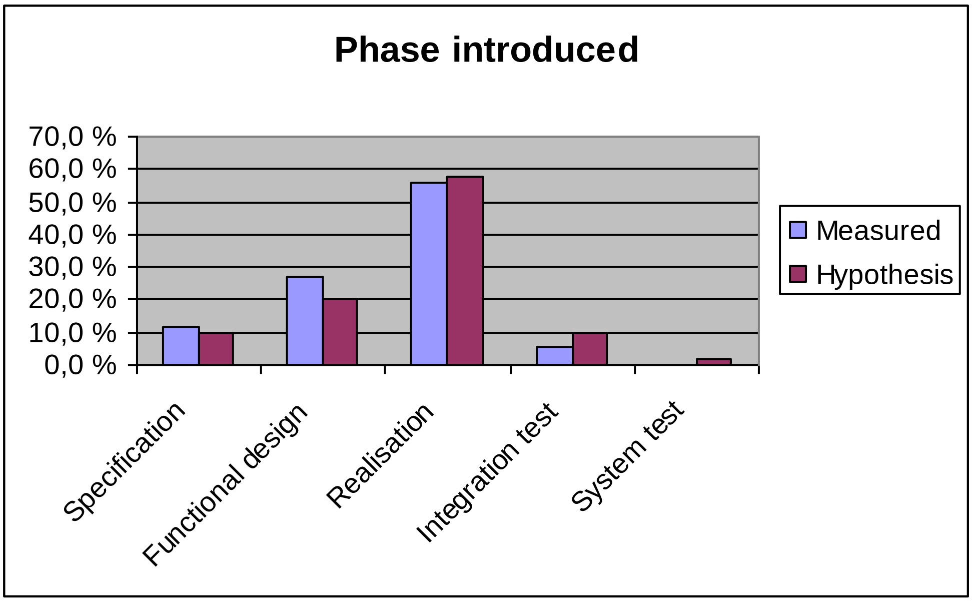


Fig. KEC_RHW 5: Errors distributed on phases

The error is per definition introduced in the earliest document that has to be corrected or changed. If a software developer has misunderstood the specification, then it is the specification that is wrong, because we have a review rule that says that a document shall not contain possibilities for misunderstanding.

The figure above shows that no errors have been introduced in the system test phase. This is because the system test has not started at the time of writing. The integration test has not finished either, so we have to wait until the end of the system test phase until we get the exact answer on this question (Q1 in the GQM abstraction sheet).

The figure above also tells us how well we know our own development process.

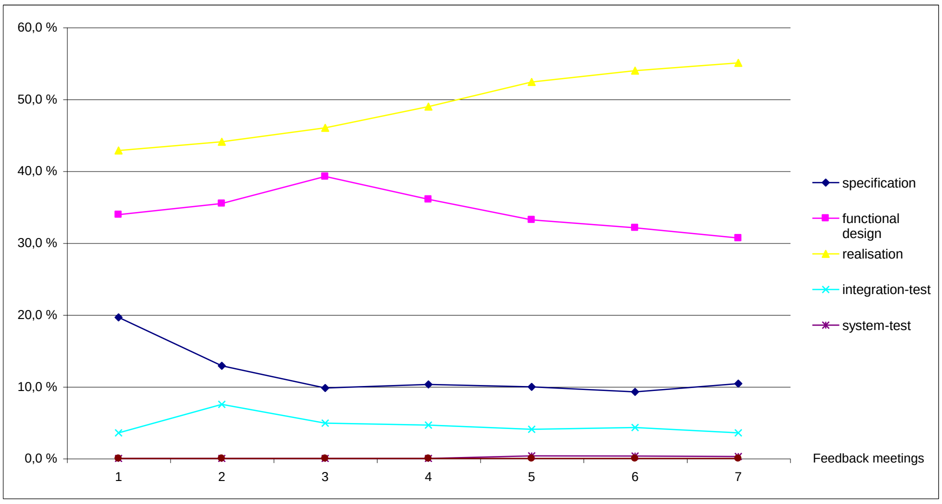


Fig. KEC_RHW 6: Accumulated distribution over time

This figure tells us that errors distributed over phase have become stable over time and can be used as a baseline.

Why are the errors introduced?

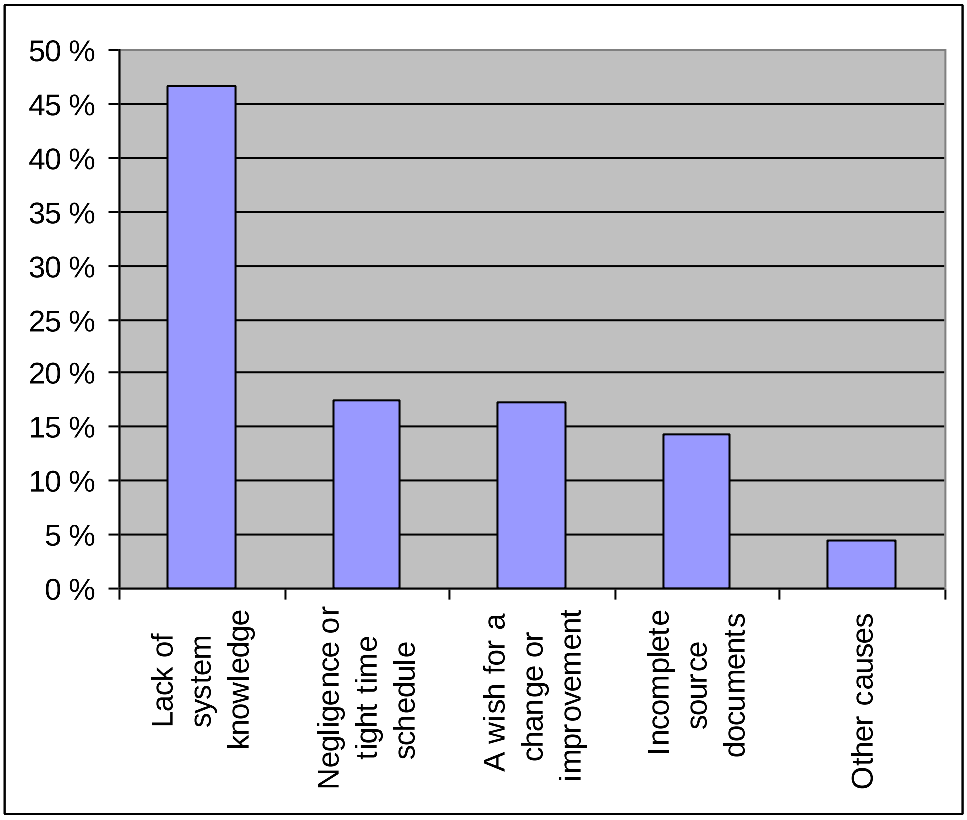


Fig. KEC_RHW 7: Why are the errors introduced (the cause)?

Maybe the most important question is to know the cause of the errors.
The most frequently cause is the **lack of system knowledge**.

Actions:

One action was therefore to check what the cause *lack of system knowledge* really means. What kind of competence is insufficient?
A request scheme was distributed among all software developers.
The following answers were the most common:

- Lack of understanding of how a function is distributed in the system/product
- Lack of understanding of the function to be implemented
- Lack of skills about the building system
- Lack of competence of software analysis/design

Out of these answers, the following actions are/will be taken:

1. Joint seminar with the system group, to understand the functions better
2. Internal course in building systems
3. The software group shall increase their competence in software design

Another cause we checked closer was the cause **lack of or poor source documents**. We wondered; which kind of source documents is poor or lacks information? Are these documents written by us (the software group) or by the system group? When we study the errors, over 60% of the errors were introduced in phase 3, which is the functional design phase. 22% belonged to the specification phase (phase 2). Additionally, the distribution of the error categories showed that most of the errors were related to sequence diagrams.

Examples of actions identified:

1. Describe and improve the written procedures in phase 3
2. Improve the procedures, guidelines, templates and checklist for writing Message Sequence Charts

What is the distribution over error categories?

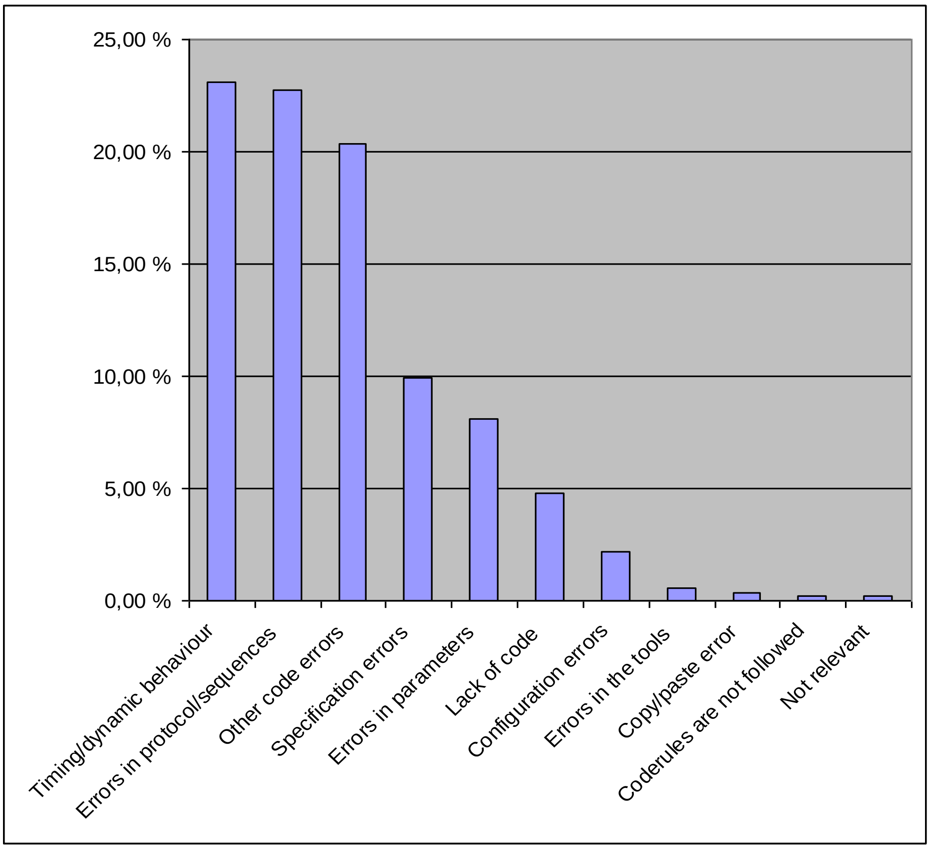


Fig. KEC_RHW 8: Errors distributed on categories

Actions:
One thing that is noticed is that the category “other code errors” is very large. This category was meant to be a small omnibus item. We thought that the categories would cover most of our faults, but obvious it doesn’t. The action will therefore be to reconsider the categories.
Beside that, the category “errors in protocols/sequences” and “faults in parameters” covers 31% of the faults. These are faults made in Message Sequence Charts and Signal Survey. The action is therefore to improve the quality of these two document types by using a new tool (Telelogic SDT) and to learn the MSC standard.
Another action is to improve the specification documents. It is the system division that writes the specification documents, so we must appeal to them on making better specifications. The action for the software division was to organise a seminar with the system division to obtain a mutual understanding on each other’s need.

The seminar between the software division and the system division, cover at least two needs: information about the functions in the radio and an understanding for the need of better source documents (seen from the software developers).

Lessons learnt

We have not succeeded answering question Q3: What is the cost of the errors distributed to the different phases and products?

It seems to be very difficult to identify the cost of the errors. Some reasons of that are:

- One symptom → more causes
- One error → more corrections, i.e. more developers involved
- Some of the developers are working in another company and are not involved in the SPIQ project

Another important aspect is the human part of it. The feedback sessions are therefore a very important part of the GQM method. Without them it wouldn't work. In these sessions the results are presented to all the developers. New actions may also come up during the feedback sessions, but most of all the feedback sessions are needed to motivate the developers to fill in the error reports with correct and complete data.

Using the Baseline Hypothesis in the GQM abstraction sheet was very useful. As you can see from the abstraction sheet, we did this on one question, *in which phase are the errors introduced?* There were very interesting discussions among the developers, and we didn't stop the discussion until everybody agreed about the answer. Because of all the discussion, the developers were very curious about what the correct (measured) answers were. This was a motivation factor for the developers when writing error reports.

Conclusion

One of our goals was to define a baseline for further improvements. We have registered the results over time, and it seems stable enough to be used as a baseline. If we want to compare the results of this project with another project, we have to take into account the surroundings i.e. the complexity of the product and the skills of the software developers. It requires that the development process (for instance the review process) is the same.

GQM is a good method as a basis for improvements. To decide the priority of actions, Pareto analysis is being used.

We had to adjust the goal and some metrics (specially the error categories and the error causes) during the collection of data.

We have already achieved positive results. One action early in the project was to improve the module test and enforce code reading (many errors were introduced because of carelessness). After three months we could see that the number of errors with this cause was reduced.

One Improvement Process as a Basis for General Process Improvement

So far we have only registered and analysed errors detected in the test phase (integration test and system test) for one project. The result is the first baseline for measuring improvements of the software process. We are now working with improvements of the software process in general, based on the results so far. At the same time we consider improvements of the metrics. Furthermore, the other software group is adopting the established system. This department has already got the results from its first analysis meeting.

We are also working with expanding the system to all phases of the development process: specification, function design, realisation, integration test and system test. In earlier phases the ‘errors’ are comments from formal reviews of documents. We have established a checklist to be used during preparation and review of documents. The reviewers’ comments have to refer to the corresponding checkpoint in order to be accepted. The data from the reviews are collected in a database to be used for analysis and decision of actions to improve the development process or the review process. The system includes the identification of the development phase in which the errors are introduced. Needs for improvement will therefore be detected for all development steps. We do not yet have any results from this expansion. However, training courses with special focus on the checklist have already been completed. The first results are expected before the end of this year.

References

[1] Basili, Victor R., *Software Modelling and Measurement: The Goal/Question/Metric Paradigm*. University of Maryland Technical Report UNIACS-TR-92-96, 1992

[1] EFQM, *Self-assessment Guidelines,* 1995

Kongsberg Ericsson Communications ANS

Staff: 150
Turnover: 300 mill NOK

Kongsberg Ericsson ranks among the world leaders in the area of mobile communication systems for military use. The company's tactical communication system, EriTac, provides communication for data and voice in tactical environment, and is supplied to both army and air defence programmes across the world. The company is co-operating closely with Ericsson companies on product development and export marketing.

Tactical Communication System

The area trunk communication system is designed according to user requirement set by the Norwegian Army and the specifications recommended by EUROCOM. Operational requirements specified therein call for demanding network functions to ensure interoperability between the tactical networks of allied nations as well as a high degree of mobility and survivability.

The system's modular design allows the user to put together any battlefield communications system, from a small point-to-point system up to a large mesh network.

Rolf H. Westgaard was born in 1938 and became M Sc. at the Norwegian Institute of Technology, University of Trondheim (NTNU) in 1964. From 1965 to 1979 he worked at Tandbergs Radiofabrikk in Oslo as radio designer, the company's education and training manager, ending up with 2 years as technical manager at their British subsidiary in Leeds.
He joined Elektrisk Bureau as a manager in the Quality section in 1979 and has since 1985 worked as a Quality Manager of the part, which ended up as Kongsberg Ericsson Communications ANS after several changes of name and ownership.
He was part of the team who started the software group of the Norwegian Society for Quality in 1983. He chaired the second European Conference of Software Quality in Oslo 1990. Rolf is responsible for the SPIQ activities in the company.

Solveig Gustad was born in the summer of 69 and became M.Sc. at the Norwegian Institute of Technology, University of Trondheim (NTNU) in 1994. Her experience is from Kongsberg Ericsson Communications ANS. The first two years she worked as a software developer and then as a software manager. As a software manager, one of her main responsibilities is to keep the development process for software as good as possible all the time. An important part of this is continuous improvements. Solveig has been involved in the SPIQ project from the start.

Key Success Factors for Business Based Improvement

Miklos Biro
Sztaki, Budapest, Hungary
Richard Messnarz
ISCN, Dublin, Ireland & ISCN Regionalstelle, Graz, Austria

Introduction

This paper is based on the results from the EU Leonardo da Vinci project PICO and discusses software process improvement from the business manager's viewpoint. Thirty experts across Europe with different backgrounds (process analysis, goal analysis, measurement, experimental approaches, business analysts) from ESPRIT projects running in the 90's have contributed to the work of PICO [3]. The result is a framework into which all methodologies fit describing how they can be combined to achieve the business manager's goals and perceptions. Experiences and case studies have been contributed from Alcatel, Siemens, Festo, and many smaller and medium sized firms from 11 different EU countries. Specifically the paper discusses key topics to be taken into consideration : business strategies, process and business indicators and their relationship, goal analysis and measurement, people factors, and infrastructure issues. The establishment of a quantitative feedback and learning cycle is emphasized.

The PICO Book:

R. Messnarz, C. Tully, (eds.), Better Software Practice for Business Benefit, IEEE Computer Society Press, Tokyo, Brussels, Washington, June 1999, ISBN : 0-7695-0049-8. It has been written by 30 authors from 11 EU countries.

A Business Driven Quality Definition

There are many definitions of quality which are extensively discussed in textbooks. Here we only mention two of them to show the slight difference between them in technical vs. business orientation.

Quality. totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs. [ISO 8402:1994, adopted in ISO/IEC 9126]

Quality. The degree to which a system, component, process, or service meets customer or user needs or expectations. [IEEE-Software Engineering Standards]

It was also recognised that in the software development process, there are several stakeholders who have naturally different views of quality. Three views are identified in the ISO/IEC 9126:1991 standard. Even though these views are incorporated in a more comprehensive quality model in the currently draft version of the new ISO/IEC 9126 standard, we prefer to present the original definitions below.

"**User's view...** Users are mainly interested in using the software, its performance and the effects of using the software. Users evaluate the software without knowing the internal aspects of the software, or how the software is developed...."

"**Developer’s view.** The process of development requires the user and the developer to use the same software quality characteristics, since they apply to requirements and acceptance. When developing off-the-shelf software, the implied needs must be reflected in the quality requirement...."

"**Manager's view.** A manager may be more interested in the overall quality rather than in a specific quality characteristic, and for this reason will need to assign weights, reflecting business requirements, to the individual characteristics.

The manager may also need to balance the quality improvement with management criteria such as schedule delay or cost overrun, because he wishes to optimise quality within limited cost, human resources and time-frame."

[Excerpts from the text of the ISO/IEC 9126:1991 standard on Information technology - Software product evaluation - Quality characteristics and guidelines for their use.]

Our objective in this paper is to elaborate on the last one of the above views.

A business manager sees process improvement or the achievement of quality under a different perspective. He wants to generate business, secure markets, increase shares, and get new contracts. Thus his interest is to create economic feedback loops that allow an increase of business potentials by investment into process improvement.

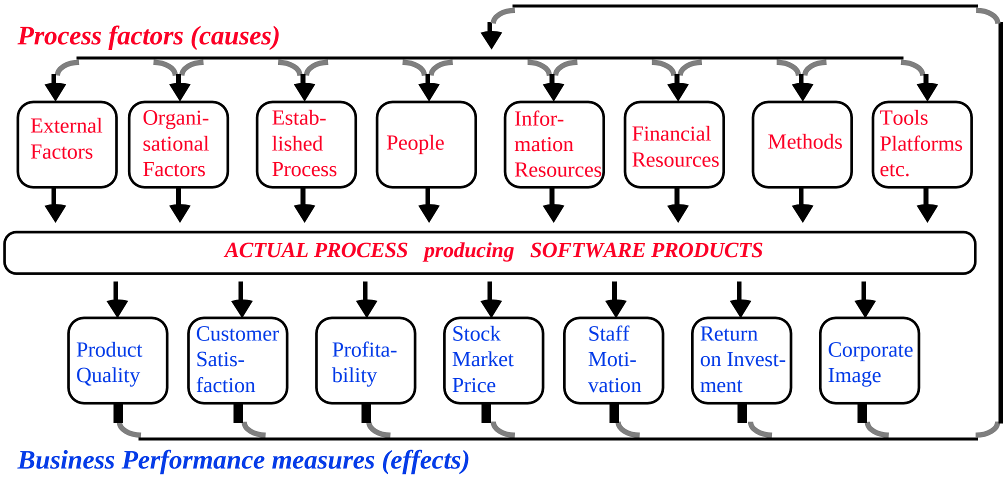


Figure 1 : A Process – Business Feedback Loop

As it is illustrated in Figure 1 investment is possible on single process factors or a combination of them, and the change in the actual design/production/delivery process will impact a set of business factors.
The business manager's greatest burden to solve is to find a traceable feedback relationship between process factors and business performance factors which will pay back.

Another important aspect is that the market (in a business environment) sometimes does not decide by actually measured quality, sometimes the perceived quality of the customer counts.

An example is the strategy of some Japanese firms to enter the European market, as it is perceived by the European competitors. A European firm (a radio manufacturer, also developing the software for radios) always was confirmed to produce radios which are running through extensive tests and thus have to be sold at a certain minimum price. The Japanese competitor realized that there is a market demand for different radios (at different price and quality levels) and started to develop and distribute radios also at low cost (with lower quality and functionality). However, not the quality of the system but the perceived quality from the different levels of the customers was deciding the market success, so that many customers bought the low price radios and the European radio sales became smaller. This does not mean that the Japanese firm was not offering quality, it rather means that they offered different systems with different quality levels (of course, also including systems of the same high standard than the one offered by the European firm).

This then comes to new success factors like flexibility, and configurability which have to be combined with the standardisation approaches underlying the process improvement models.

And in a typical business scenario thus a division could produce 100% quality and achieve high organisational maturity grades, but it still could fail in business due, for instance, missing flexibility and system configurability.

Another example is the Microsoft and Apple story. A major reason why the quality of a MaIntosh is perceived (by the customers) as higher than that of a

PC is the fact that a Mac (once bought) kept stable over years, whereas due to a upgrade philosophy of rapidly changing versions with exponentially increasing resource demands the PCs are just stable for about 1.5 years and then have either to be largely upgraded or exchanged.

Now imagine a situation in which the same game starts with the Mac, where due to largely increasing resource demands also the Mac has to be upgraded every 1.5 years. This would lead to a loose of the stability argument and the customers would perceive the quality of PC and Mac as the same, thus buying the low cost PCs in the future.

Of course, this is just a story, but it shows some ideas of typical business strategies, which at the end calculates not only with defect rates but with perceived quality from customers.

Business Manager's Quality Perception. A business manager invests into processes, people, and infrastructure with the aim to ***satisfy market demands and perceived quality from customers***, with a view of creating a traceable process – business feedback loop.

Below, we present an original Structured View of Business Motivations for SPI first published in the PICO book.

A Structured View of Business Motivations for SPI

One of the most pertinent questions a business manager can ask is the following "How can I make my firm succeed where another fails?" Managers with financial, operating, production, marketing, human behavioural, or other orientations will give a variety of answers to this question and will arduously argue for their valuable ideas. Here, we will outline a framework integrating and structuring several orientations [4], [5], [6], [19].

The key concept of the approach is the notion of lever. Levers are means used by a firm to increase its resource generating ability, just as a mechanical lever is used for increasing the force applied to an object. The analogy goes even further. Just as a force can be applied in many different ways to the object resulting in a similar displacement, the use of the different levers can increase the resource generating ability of the firm resulting in similar business benefits. Finally, the resources are used to increase the assets of the firm and to reward employees and stockholders.

Let us analyze the ways software process improvement can provide leverage to a firm from the financial, operating, production, marketing, and human behavioural perspectives.

Financial leverage

Financial leverage means borrowing funds and investing them with a return higher than the cost of the debt. If a company is able to exploit financial leverage, it can make money on funds it does not own. Can software process improvement provide financial leverage to a firm? The answer is clearly yes if the return is high enough to make it worth borrowing money for achieving it. In other words, is the ROI (return on investment) for software process improvement high enough?

This issue is discussed in more detail in the PICO book. Here, we only mention a few determining numbers which allow the reader to form an idea about the magnitude of the leverage that can be achieved. ROI is considered as "the bottom-line figure of most interest to many practitioners and managers" in a pioneering report [14,15] of the Software Engineering Institute of Carnegie Mellon University. The value of this report lies in the fact that it contributes to the satisfaction of the major need of companies for quantitative information regarding the benefits of SPI before committing resources and investing into it. There were 13 organisations, where CMM® [1]-based SPI [17] occurred prior to 1990, which agreed to provide their highly sensitive and confidential data to the SEI for anonymous or identified reporting according to their own decision. The companies were the following:

- Bull HN
- GTE Government Systems
- Hewlett Packard
- Hughes Aircraft Co.
- Loral Federal Systems (formerly IBM Federal Systems Company)
- Lockheed Sanders
- Motorola
- Northrop
- Schlumberger
- Siemens Stromberg-Carlson
- Texas Instruments
- United States Air Force Oklahoma City Air Logistics Centre
- United States Navy Fleet Combat Direction Systems Support Activity

The report does not claim that its results are typical. It only shows the potential benefits of SPI in a favourable environment. And these benefits in terms of ROI are more than impressive. The range of the ratio of measured benefits to measured costs is between 4x and 8.8x over periods of software process improvement ranging from 3.5 to 6 years. Benefits include savings from productivity gains and fewer defects, but do not include the value of enhanced competitive position which will be examined below under the title

[1] ® CMM is registered at the U.S. Patents and Trademarks Office. CMM^{sm} is a serviced mark of the Carnegie Mellon University.

marketing leverage. Costs include the cost of a Software Engineering Process Group (SEPG), assessments, and training, but do not include indirect costs such as incidental staff time to put new procedures into place.

Capers Jones [9] reports an ROI of 3x to 30x with the returns measured over a 48 months period using the Software Productivity Research (SPR) assessment method and baseline studies. Exceptionally in the literature, Capers Jones, [9] also reports a negative and alarming record: “Several companies and government agencies have managed to spend in excess of $10,000 per capita with no tangible benefits accruing.”

The conclusion of this section is that yes, software process improvement can provide significant financial leverage to the firm making it worth borrowing money for investing into it. Nevertheless, the effect can be adverse if the company does not pay appropriate attention to accrued costs and to immediate exploitation.

Operating Leverage

The profitability of a firm highly depends on its cost structure, that is the repartition between its fixed costs and variable costs. Operating leverage means the relative change in profit induced by a relative change in volume, which is clearly higher for a firm with lower variable costs. Nevertheless, the achievement of a low variable cost production usually presumes high fixed costs, that is a capital intensive process.

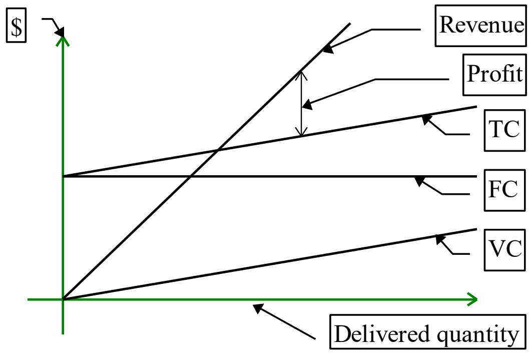


Figure 2 : Firm with high fixed costs (FC) and low variable costs (VC). The total cost is TC = FC + VC.

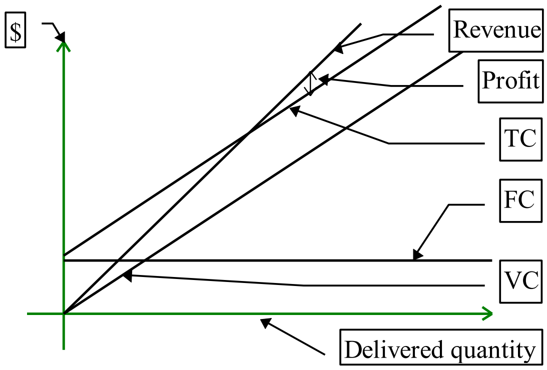


Figure 3: Firm with low fixed costs (FC) and high variable costs (VC). The total cost is TC = FC + VC.

Software process improvement clearly means an increase in fixed costs, which include training, consulting fees, equipment, software licenses and improvements in office conditions. However, the question is whether the company is really able to use it for decreasing its variable costs. Measuring the variable costs of software production is not a straightforward issue. The notion of function point had to be invented to resolve this problem among others. Function point analysis is discussed in more detail in chapter 6 of the PICO book.

Function points are results of well defined calculations based on different characteristics of a software product that are of interest to users: inputs, outputs, data groups, inquiries, interfaces. The cost of the average number of person months necessary for delivering a fixed quantity of function points is a sample measurement for the variable costs of software production (person months/function point). However, the measurement mostly used in reports on software process improvement is development productivity (function points/person month) which is in fact the reciprocal of the above number. In the following, reported results are presented in terms of development productivity whose increase is consequently equivalent to the decrease of the variable cost of software production.

If, due to software process improvement, a software firm is able to deliver the same quantity of function points using less person months than its competitors, then it will have the potential to take advantage of operating leverage. Nevertheless, real profit will only be generated if the revenue resulting from actually delivered function points exceeds the total of the cost (TC) of software process improvement (FC) and the cost of person months used for generating them (VC). This means that bigger firms with a larger number of delivered function points will have a better chance to enjoy the operating leverage resulting from software process improvement.

Productivity gains per year measured in lines of code (LOC) per unit of time are reported in [14,15] to be between 9% and 67% at the examined organisations. Another form of productivity gain particularly relevant to software is due to the earlier detection of defects also presented in [14,15]. The figures show a 6-25% increase in the number of early detected defects. This represents enormous savings if we consider that “the cost of fixing a defect pre-release is approximately $50 per line of code, while the cost of fixing a defect discovered post-release is about $4000 per line of code”.

It is important to highlight at this point that the major European company SIEMENS is a world-wide pioneer in measuring and publishing information related to productivity gains resulting from software process improvement. They report the following experimental reductions in error costs based on maturity levels [25]:

- 17% from Level 1 to Level 2,
- 22% from Level 2 to Level 3,
- 19% from Level 3 to Level 4,
- 44% from Level 4 to Level 5.

SIEMENS also reports productivity increases in terms of lines of code, but at this point it is more appropriate referring to chapter 8 of the PICO book giving direct account of SIEMENS experiences.

Another report accounting for productivity gains due to process improvement in European software development organisations originates from the results of a questionnaire developed by IBM Europe [13]. This report compares, among others, the performance of leaders with that of laggards from among 360 responding organisations from 15 countries. While leaders "achieve a development productivity of 25 function points per person month; remove over 95% of defects before delivery; estimate consistently to within 10% of the actual cost and duration of a project; and spend less than 1% of the development effort on defect correction in the first 12 months after delivery", laggards "have a development productivity below 5 function points per person months; remove less than 50% of defects before delivery; have projects which often exceed estimates by more than 40%; and spend more than 10% of the development effort on defect correction in the first 12 months after delivery".

	leaders	laggards
Development productivity (function points/person month)	25	<5
defects removed before delivery	95%	<50%
inaccuracy of cost and duration estimates	10%	>40%
% of development effort spent on defect correction in the first 12 months after delivery	<1%	>10%

Table 1 : Data from Leaders and Laggards

Production Leverage

Production leverage is the rate of growth of profits resulting from cost declines due to the accumulation of production. It is an empirical fact that unit production costs decline exponentially when experiences are accumulated and the steady reuse of these experiences is well managed by the firm.

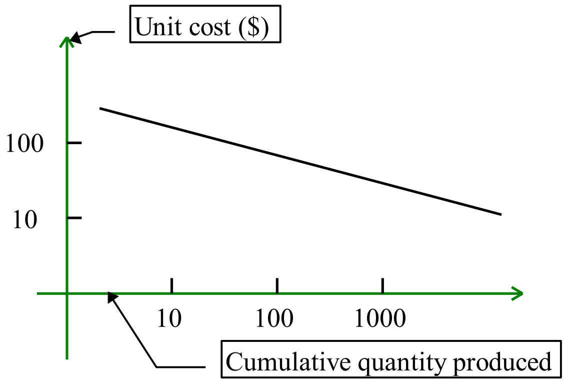


Figure 4 : The Experience Curve in a logarithmically scaled system of coordinates

The graph of the unit costs in function of the cumulative quantity produced is called the experience curve which is usually represented as a straight line in a logarithmically scaled system of coordinates. The existence of the experience curve is essentially due to economies of scale, learning, improvements, and reuse.

The accumulation of experiences and the management of their steady reuse is clearly one of the primary objectives of software process improvement. Interestingly, this aspect of software process improvement has not been analysed directly. Nevertheless, the paper [14,15] acknowledges that the techniques useful for tracking the cost changes over time do not specify what is causing the changes. The cause may be the process improvement, but it may also be increased experience, new analysis and design methods, new tools, and so on.

Marketing Leverage

Marketing leverage means the effect of higher prices and innovative distribution on profits. Software process improvement, maturity achievement, ISO 9000, or TickIT certification have an important impact on the perceived capability of the company and on the perceived value of its products, which contributes to improved customer satisfaction and makes it possible to achieve higher prices.

Quality and process improvement are part of a differentiation strategy in which the business delivers and is perceived to deliver a product or service superior to that of competitors. In a study of 248 distinct businesses in the service and high-tech industries referred to in [1], “a reputation for quality was the most frequently mentioned sustainable competitive advantage”.

In line with the above US study, a major European company, Lloyds Bank Plc. lists the demonstration of competitiveness through CMM/SPICE/TICKIT certification as one of the key drivers for software process improvement [18].

The experiences of SIEMENS, another major European company, were already mentioned and are described in more detail in chapter 8 of the PICO book. The “promotion of the external visibility of Siemens’ software competence” is listed as an important area to focus [25]. They also report that “highly-predictable quality regarding system releases and costs led to greater market acceptance”.

[14,15] acknowledge the importance of “improved reputation, good will, and brand name recognition” as intangible benefits of process improvement arising from the “impact of SPI on customers”, but they present no actual results relating to these.

A survey reported in [10] provides the feedback of more than 50 companies on the benefits gained from the TickIT certification scheme. One of the major benefits is formulated in the following way: “Customers have increased confidence in the quality of our products. With the advent of TickIT the UK Ministry of Defence (and many other influential purchasers) have ceased their second party assessment activity, while many other large organisations now insist that their suppliers and product resale partners achieve ISO 9001 certification by a TickIT accredited certification body.”

One of the rare reports which provide a measurement of the direct effect of software process improvement on the marketing leverage of a company was presented by Peterson [24]. The report is based on 560 SEI software process assessments through December 1995 whose results were provided to the SEI by March 1996. The statistics give the percent of respondents reporting “excellent” or “good” customer satisfaction when improving software processes from the initial level (around 80%) through the repeatable level (a surprising decrease to 70%) upto the defined level (around 100%).

Human Leverage

Human leverage means the effect of employee motivation on profits. It is widely known that employee motivation (empowerment) can be significantly influenced by immaterial means like management styles and organisational structures. Huge individual energies can be released for example in an appropriate teamwork environment where team members are simply given the responsibility to do their jobs as well as they can, instead of exerting close surveillance over them. Nevertheless, attention must be paid at the differences in the collective mental programming of people in different national cultures [12].

The exploitation of human leverage is particularly important in software process improvement since software development is a fundamentally human mental process.
Herbsleb & al. [14,15] classify the “impact of the SPI effort on the organisation’s employees”, including “better morale, improved understanding of the corporate mission and vision, fewer crises, less stress, less turnover, and better communication within the organisation”, among the intangible benefits of SPI. Actually, no measurements are presented relating to these benefits in this report.

The already mentioned report presented by Peterson [24] also shows statistics giving the percent of respondents reporting “excellent” or “good” staff morale when improving software processes from the initial level (around

20%) through the repeatable level (around 50%) up to the defined level (around 60%).

There is an already mentioned important study, [13] initiated by IBM Europe, which gives a measurement of the impact of employee morale on the level of performance of a company. The statistics based on 360 responding organisations from 15 European countries show that employee morale correlates strongly with both delivery performance and quality performance levels.

An ultimate recognition of the importance of human leverage is the development of models directly addressing this issue: the People Capability Maturity Model (P-CMM) [11], and the Personal Software Process (PSP).

Application of Principles

When PICO started in 1995 each of the different methodologies argued to be the best, to insert the most important rules and principles. But over the project time and comparing different industrial case studies it turned out that you are most efficient if you take into account all principles from all methodologies.

It is equally important (as stated in [19])

- To establish a business context
- To translate the business goals into technical goals, and establish quantitative relationships which allow to track if the technical goals of process improvement are met and how they contribute to the achievement of business goals (see GQM, and ami [2])
- To analyse the strengths and weaknesses and identify which improvements would show the highest impact on the increased potential of achievement of business goals (see PICO book chapter 1)
- To follow technical guidelines to achieve higher levels of organisational efficiency (see the maturity models as offered by the different assessment providers, with the most well known one being SEI/CMM [17])
- To establish frameworks which allow people to increase their skills and potentials in alignment with the organisational improvement changes in the company.
- To work towards an electronic solution in which process improvement and quality is an integrated part of each work place within the organisation, supporting team-work across offices and partnerships.

Goals and Measurement

A big electronics company in Europe [19], for instance, made an assessment resulting in maturity levels for different areas of the organization and the identification of weaknesses such as unrealistic planning, no process for design reviews, and weak configuration management. The organization was already ISO 9001 certified but only 30% actually accepted the guidelines due to missing practicability (formally good documented but not realistic for projects in the field). A formal pragmatic assessment and improvement approach would then, for example, decide about introduction of configuration management and so forth, BUT does this really now meet the organization's business goals?

So this electronics company decided to run a goal analysis (based on the GQM approach) in parallel interviewing business managers, department heads, IT managers and project managers and designing a consistent goal tree from top to bottom.

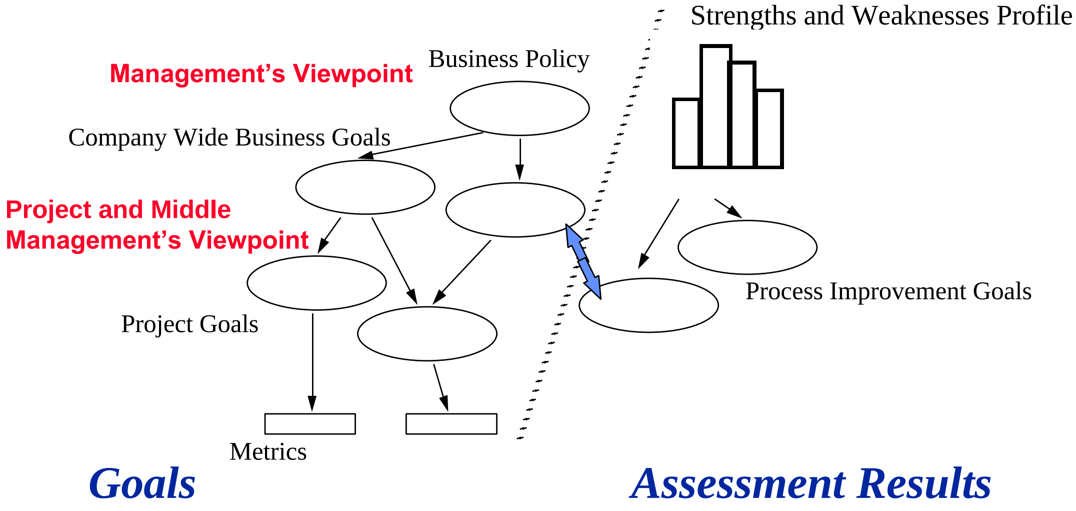

Strengths and Weaknesses Profile
Management's Viewpoint
Business Policy
Company Wide Business Goals
Project and Middle Management's Viewpoint
Project Goals
Process Improvement Goals
Metrics
Goals
Assessment Results

Figure 5 : Aligning Business Goals [21] with Improvement of Weaknesses Found in Assessments [20]

One of the specific business policies was to create a financial framework for the next years which allows to get a reserve budget to fight for a brand new product on the market. To get to this marketing budget the business managers decided to stabilize the development effort from divisions at x% so that with all other overheads and cost a certain percentage is saved every year to have the budget together right at the time when the product is announced. At this moment the divisions were certainly higher than x% and the improvement actions (based on the previously identified weaknesses from the assessment) had to demonstrate after 3 years the success of achieving this business goal.
The technical staff were frightened and thought that people will be dismissed but the truth was that a proper interpretation of the business goals led to a completely different view. The business managers expected that process improvement provides a better work process and environment so that with the same staff more projects and tasks can be done and over time the development effort is stabilized at x%.
Under this perspective the 3 process weaknesses were again analyzed and further interviews showed a potential of re-use because in all systems in the sector nearly 80% of the functionality was always the same. So the improvement plan focused on an integration of design, configuration management and review of a re-use pool of these 80% functions and to reduce the development for each project to the only 20% additions, thus enhancing productivity and achieving the effort stabilization.

Now, let us assume that only a pragmatic assessment would have been performed. Three weaknesses would have been identified and without re-use orientation would have led to a pragmatic proposal to first run a pilot project to identify a configuration management system and field test it, to disseminate it to other projects, and to help making it a division wide standard. Sounds simple, BUT unfortunately the business context is lost. What then happens is that management sees additional effort, the development effort further increases, and with no vision of decrease of the development effort the business manager after 1 year (before benefits can be made visible) would really decide about things like dismissals.

This example refers back to the ***principles of operating and production leverage***.

Processes and People

Business managers, project managers, and practitioners speak different languages and might have different viewpoints on the same situation [22,23]. Business managers speak about fixed cost, variable cost, return on investment, leveraging, market trends, product sales, and customer satisfaction. Middle and project managers speak about budget, work plans, quality plans, configuration management, requirements analysis and structured analysis, and always fear to be delayed or to overrun the budget provided by the business managers. Practitioners deal with modules, design them, implement and test and deliver them so that they can be integrated into the system architecture planned by the project manager.

It is the nature of process improvement methodologies that measurement and control functions are installed which again will be seen differently from the different target groups. Business managers not understanding that SPI needs investment with a ROI in about 3 years sometimes demand that process improvement is performed without any assignment of budget to it: lets do quality but it should not cost any dollar. This certainly leads to a disaster and top management commitment is the number one success criteria for starting an improvement program. Middle managers will like the process improvement most because it provides them with methodologies and facilities to better define the processes, to better visualize the productivity and quality, and to improve the predictability which leads to the fact that schedules and budget are kept satisfying therefore the business managers. At the beginning the practitioners usually see the implementation of a process improvement program as a dirty trick of middle and project management to better control their performance.
However, after some time they start to realize that more reliable plans give them enough time for design, better design reduces the re-work and maintenance stress. Formalized reports help them to identify the root cause of problems and to track the correction, and they can learn and improve themselves based on measures [22,23].

It is a key to success to have all groups behind the initiative and to act as a translator of the different viewpoints.

For instance, in a European project Bestregit (which applied PICO principles in general management and technology transfer institutions) role based teamwork models were established for best practices, and an experiment at a Spanish site showed that using these role based models leads to

- People ***identifying themselves*** with roles in a team
- A 2/3 ***effort reduction*** in the introduction of new staff

For instance, PICO principles were applied for ISO 9001 certification based on computer supported team-work scenarios (tried out with certifications at TÜV, ÖQS, and Norske Veritas). Measurements at Austrian sites showed that the actual ***motivation*** of engineers to use the standards grew from 17% to 57% of the staff.

These effects relate back to the ***principle of human leverage***.

Integrated Team-work and Infrastructure

On an Intranet Scale –

A major fear of engineers when new standards are introduced is the additional expected amount of documentation.
In old traditional situations of standard introduction a process group is established which creates and maintains the standard, produces a manual, and looks that all projects keep the guidelines described in the manual.
However, engineers do not like to see quality or improvement as a separate part in parallel to their normal engineering work. Quality and improvement related processes should be an integrated part of their work place, just like a compiler is.

This leads to Intranet based systems [22] that support

- Teamwork and information flows
- Information structuring and archiving
- Joint use of resources for design and implementation
- Configuration management and version control
- Etc.

European projects have tested such environments in companies and found, for instance [5], a number of key factors:

a. If (as said under People and Processes) role centred work models better support human leveraging, it is important to create infrastructures that allow to establish role based team-work over the net
b. If a system should be able to work for different departments and processes with the same software, the system must be generic and highly configurable.
c. Especially guidance in documentation through the system should be supported to such an extent that the additional expected documentation (the engineers' fear) gets minimised, although a full implementation of the standards is achieved.

On a Global Scale -

To stay competitive on the global market it is necessary to set up win-win based agreements in cost sharing projects in which partners from different countries share the risk and the effort and jointly exploit ideas, products, and services. Through effective and distributed collaborations organisations can cut down their risk significantly (e.g. sharing the development cost with other partners) and can reach a much larger market [22].

However, the key problem is that distributed collaboration needs effective co-ordination of the work of the different partners. And old conservative means such as direct supervision, local meetings, large local and not distributed teams, do not work any more. The decomposition into smaller competence teams with clear cooperation interfaces supported by new and effective communication systems is needed. This includes a virtual office on the net with project archives and document management, configuration management, guide-lines and computer support for project documentation, network and computer supported information flows, and appropriate security mechanisms assuring privacy of the materials exchanged and produced.

A field test implementation of such a system at different companies is described in [22]. Companies using the system described in [22] have successfully achieved ISO 9001 certification based on fully computer supported team-work processes.

This new approach of collaborative development leads to a big chance for creating financial leverage (by joint risk and effort funding) and an increased marketing leverage (by joint representation on the market, and larger distribution through a net of partnerships).

A Comprehensive Definition of SPI

SPI [19] is a strategy for business managers to align their business goals with technical improvement objectives, to apply a set of different methodologies

- Assessment methods
- Goal Analysis Strategies
- Measurement Tools
- Paradigms and Strategies to Establish People Motivation and Team-work Processes
- Improvement Planning Techniques

to create a process – business feedback loop, with the aim to make the organization more competitive (*Marketing Leverage*). Business orientation is a must to create the budget for SPI (*Financial Leverage*). People management and motivation is a must to get a critical mass of people following the SPI vision (*Human Leverage*). Goal trees are a must to translate the business manager's viewpoints into practical objectives for the SPI teams (aiming at *Production and Operating Leverage*). And pragmatic assessment methods (CMM, Bootstrap, TickIT, Trillium, etc.) are just one tool to evaluate the strengths and weaknesses before applying the combination of the other approaches (from business orientation to goal analysis).

PICO's Outlook

PICO produced a book [19], and a set of training courses for

- Business strategies
- Goal analysis
- Process analysis
- Measurement
- Experiences
- Self Assessment

PICO also developed a tool set which can work (by configuration) with almost all different assessment methodologies. This was required because PICO contributors came from many different methodology backgrounds.

PICO did not develop a new methodology, it is rather a collection of experiences of how to combine existing approaches to achieve a business based strategy.

PICO's book (although announced earlier already) is still at IEEE in the production, and when the book comes out PICO will distribute at moderate prices a CD with all courses, the tool, and further SPI materials as a complementary set to the book.

References

[1] Aaker,D.A. Strategic Market Management. John Wiley & Sons, Inc., 1995.

[2] The ami Handbook, 1995, ISBN 0-201-87746-5

[3] Biró,M.; Feuer,É.; Haase,V.; Koch,G.R.; Kugler,H.J.; Messnarz,R.; Remzsö,T. BOOTSTRAP and ISCN a current look at the European Software Quality Network. In: The Challenge of Networking: Connecting Equipment, Humans, Institutions (ed. by D. Sima, G. Haring). (R.Oldenbourg, Wien, München, 1993) pp.97-106.

[4] Biró,M.; Sz.Turchányi,P. Systems of Decision Criteria Supporting Software Process Improvement. In: DSS - Galore, Proceedings of the fifth Meeting of the EURO Working Group on Decision Support Systems (ed. by M. Brännback, T. Leino). (Institute for Advanced Management Systems Research, Turku, Finland, 1995) pp.133-147.

[5] Biró,M.; Feuer,É.; Remzsö,T.; Sz.Turchányi,P. Business Decision Problems Supported by Software Product and Process Assessment. In: Proceedings of the ESI-ISCN '95 Conference on Practical Improvement of Software Processes and Products (ed. by T. Katsoulakos, R. Messnarz). (European Software Institute-International Software Consulting Network, Vienna, Austria, 1995).

[6] Biró,M.; Sz.Turchányi,P. Software Process Assessment and Improvement from a Decision Making Perspective. ERCIM News (European Research Consortium for Informatics and Mathematics) No.23 (1995) pp.11-12. (http://www-ercim.inria.fr/www-ercim.inria.fr/publication/Ercim_News/enw23/sq-sztaki.html)

[7] Biró,M.; Feuer,É; Ivanyos,J. Process Improvement Expriment at MemoLuX. In: Proceedings of the ESI&ISCN 1997 Conference on Practical Improvement of Software Processes and Products (ed. by R.Messnarz). (International Software Collaborative Network, Budapest, 1997) pp.6.18-6.29.

[8] Biró,M.; Remzső,T. Business Motivations for Software Process Improvement. ERCIM News (European Research Consortium for Informatics and Mathematics) No.32 (1998)

pp.40-41.
(http://www-ercim.inria.fr/www-ercim.inria.fr/publication/Ercim_News/enw32/biro.html)

[9] Capers Jones, The Pragmatics of Software Process Improvement. Software Process Newsletter. No.5, Winter 1996, pp.1-4.

[10] CSA, TickIT provides proven quality benefits to both customers and suppliers of software systems. CSA (Computer Services Association) Position Paper. Pub. No. 32, 1994.

[11] Curtis,B; Hefley,W.E.; Miller,S. Overview of the People Capability Maturity Model. Software Engineering Institute, Carnegie Mellon University, Maturity Model CMU/SEI-95-MM-01.

[12] Geert Hofstede, Motivation, Leadership, and Organisation: Do American Theories Apply Abroad? Organisational Dynamics. Summer 1980, pp.42-63.

[13] Goodhew,P. Achieving real improvements in performance from software process improvement initiatives. European Software Engineering Process Group Conference 1996 (C306).

[14] Herbsleb,Jim; Hayes,Will. Performance Profile: Measuring the Business Value of Software Process and Technology Improvements. European Software Engineering Process Group Conference 1996 (C317).

[15] Herbsleb,J; Carleton,A; Rozum,J; Siegel,J; Zubrow,D. Benefits of CMM-Based Software Process Improvement: Initial Results. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-94-TR-13.

[16] Ivanyos,J.; Biró,M.; Messnarz,R. The PASS Process Improvement Experiment in Hungary (PASS EP 21223). In: Proceedings of the EuroSPI'1998 Conference on How to reap the business benefit from Software Process Improvement (ed. by R.Messnarz). (International Software Collaborative Network, Gothenburg, 1998) pp.8.24-8.45.

[17] Key Practices of the Capability Maturity Model V 1.1, CMU/SEI-93-TR-25, Feb. 93

[18] Larner,C. More practical experiences and lessons gained by the software engineering process group of a major European bank: a question of ownership. European Software Engineering Process Group Conference 1996 (C304).

[19] R. Messnarz, C. Tully, (eds.), Better Software Practice for Business Benefit, IEEE Computer Society Press, Tokyo, Brussels, Washington, ISBN : 0-7695-0049-8, to be published.

[20] Messnarz R., et. al. , BOOTSTRAP: Fine Tuning Process Assessment, IEEE Software, pp. 25-35, July 1994

[21] Messnarz R., Kuvaja P., Practical Experience with the Establishment of Improvement Plans, in: Proceedings of the ISCN'96/SP'96 Congress on December 1996 in Brighton, pp. 155-169, ISCN Ltd. Dublin, Ireland

[22] Messnarz R., et.al., NQA – Network based Quality Assurance, in: Proceedings of the 6th European Conference on Quality Assurance, 10-12 April 1999, Vienna, Austria

[23] Thamhain H.J., Wilemon D.L., Criteria for Controlling Projects According to Plan, in (eds.) Nahouraii E., Butler J.T., Petry F.E., Richter C., Tham K., *Software Engineering Project Management*, pp. 15-54, IEEE Computer Society Press, USA 1990

[24] Peterson, B. Software Process Improvement Trends. European Software Engineering Process Group Conference 1996 (C306).

[25] Völker, A; Gonauser,M. Why Maturity Matters. European Software Engineering Process Group Conference 1996 (C305).

SPI – Why isn’t it more used?

Tor Stålhane, Ph.D.,
Kari Juul Wedde, M.Sc.,
SINTEF, Trondheim, Norway

Introduction

SINTEF has for some years offered assistance to industry in Norway and Sweden in the area of SPI – Software Process Improvement. This assistance is based on two projects, namely:

The QIS project – Quality Improvement in Scandinavia. This is an EU sponsored project that shall give help and assistance to PIEs in Norway and Sweden and market the concept of SPI to the software industry in these two countries.

The SPIQ program – Software Process Improvement for better Quality. This is a national Norwegian program that is partly a national ESSI type project and partly a co-operation between academia and Norwegian industry to increase the use of process improvement methods in Norwegian software industry.

From the start we thought that it would be an easy job to sell SPI to the industry. However, this turned out to be a rather optimistic assumption. Quite a lot of companies did not believe that SPI would solve their problems. This was a general attitude and not limited to management or developers only.
The rest of this paper describes our findings related to why a large part of the industry rejected the very concept of SPI. We start by describing the situation as we see it and the reasons that we have found, both among management and among developers. Then we sketch some suggestions for solutions in order to improve the situation before we end the paper with some conclusions.

The current situation – and how we got there

Where we got our data

We base our conclusions on several data sets. The data are mostly qualitative and not collected in order to throw light over the problem at hand. This is a consequence of the fact that we got a clear picture of the sordid situation quite late in the process. The data sources are as follows:

A survey of Norwegian companies - used to discover which factors the companies considered important for improvement 0.

A survey of European companies - used to discover why companies do not do SPI and what they do instead in order to improve 0.

A set of interviews with quality assurance (QA) personnel and developers in five Norwegian companies on the role of written procedures as a vehicle for organizational learning 0.

Data collected in two Norwegian companies pertaining to where they are and where they want to be in the near future – also called a gap analyses. This includes, among other things, also a set of arguments for and against SPI.

Discussions with developers in Norwegian and Swedish industry on SPI.

All this information gives clues to the problem, even though none of them present the full picture. Thus, instead of a data analyses – statistical or otherwise – we present a solution to a jigsaw puzzle consisting of all the small pieces of information that have been given to us.

The way we are – and why

Everything has a history, so also the current status of QA, SPI and software development. In the seventies, software development was a rather chaotic affaire. The programmers considered themselves artisans – or even artist – that did not produce, but instead create. Project control, quality control and any other control was mostly absent: The project was finished when it was finished and the customers stood in awe and watched the blinking lights on the computer.

The managers did not think this was a good state for the business and project control seemed to be a solution that appeared just in time. When it was sanctified by ISO and given an official number, the managers were all too ready to jump on the bandwagon. ISO 9000 was not a bad idea; it was just that its introduction was badly timed. Since the management layer of the companies were screaming for control over hundreds of cowboy programmers, ISO 9000 lead to an intense focus on control. As a result of this rather myopic interpretation of the standard, two things happened:

The management focused on control, since ISO 9000 promised to give them just that, and the QA department followed suite.

The most important part of ISO 9001 – section 4.14 that focused on process improvement - was lost in the introduction. The fact that it in many cases was sold as statistical process control (SPC) did not help much either.

Thus, the big chance to introduce real quality improvement and SPI into software development was lost. Instead we could hear ghastly statements from QA managers like "The QA department's most important job is to give the managers full control over the projects".

Some data – and our interpretation

In order to see if there is a way of adding more details to the picture, we have looked at two contrasts that we hope will shed light on the status concerning SPI. These contrasts are small versus large companies and the way the QA department looks at SPI versus SPI as seen from the developers.

Small versus large companies

There seems to be a larger share of the big companies that do SPI than there is of the small companies. There are at last two explanations for that:

Either: SPI will lead to changes. It is thus a risk and larger companies can better afford to take such a risk. The main reason for this is that they have more resources. In addition, SPI will take time and larger companies usually have a longer planning horizon.

Or: Larger companies need SPI, small companies don't. The reason for this is that small companies are more flexible, is less dependent on written procedures and has more efficient internal communication.

When considering SPI and small companies, there are also two other areas that should be taken into considerations. These are:

Small companies live in an ever-changing world and they have little or no influence over the way their environment changes. Thus, anything that depends on a stable environment has little or no value for a small company. Since SPI to a large extent is a synthesis of previous experience, SPI becomes at least partly irrelevant.

SPI will require a certain minimum of documented routines. However, written procedures and routines have a tendency to petrify the organization and thus remove some of the flexibility that a small company needs in order to compete efficiently.

Thus, small companies focus on being innovative and flexible, instead of using what they see as a large documentation overhead that they fear will take away all of their competitive advantages. This stance is, however, not limited to small companies. A manager from a large company that produces mobile telephones have stated that they are a CMM level 1 0 company and intend to stay there for the foreseeable future. This does not mean that they do not do SPI – on the contrary, they do quite a lot of it. The point is that they look for ways to improve themselves that do not destroy creativity and innovation.

QA department versus the developers

One of the observations that is rather consistent in all our material relates to the on-going discussion of what kind of information and knowledge that is important for a company. The programmers focus on how to write code, while the QA department focuses on rules and regulations. This was brought clearly forward when developers and QA personnel were asked about the value of procedures as a vehicle for knowledge dissemination in the company. The results are presented in Table 1.

	Procedures is an important vehicle for knowledge transfer		
Personnel	Agree	Undecided	Disagree
QA managers	64%	18%	18%
Developers	0%	50%	50%

Table 1 Procedures as a vehicle for knowledge transfer

The message is loud and clear – QA managers think that procedures, document forms etc is important knowledge, while the developers think it is not. This attitude also surfaces in another question, related to the use or no use of the company's written procedures. The developers were asked about the amount of control and the amount of use of the company's procedures. The results are summarized in Table 2.

	Use of company procedures		
Degree of control	Low	Medium	High
Low	36%	55%	9%
Medium	33%	34%	33%
High	0%	56%	44%

Table 2 Use of company procedures

It is straightforward to see that the degree of control has a strong influence on the degree of use of the company's procedures. A rather sad conclusion is that the developers do not use the procedures because they find them useful but because they get in trouble if they do not use them. A collateral to this conclusion is that the developers do not see the procedures as particularly useful. This result is consistent with the conclusions from the previous question.

If written procedures is not the answer, what is? We asked both the developers and the QA managers which alternative vehicles they would consider. As shown in Table 3, the degree of agreement between the two groups is good. We see that they agree on the most important items, which are – in order of importance:

Experience bases – saving experience for later reuse in new projects. The experience base can be at any level of formalization – from a paper archive of experience reports to a complete Experience Factory (EF).

Socializing – meeting at the coffee machine, in the cantina or somewhere else in order to exchange experiences, ideas, insights and complaints.

Study groups – a more formalized version of "Socializing".

Reports and documentation – a less formalised version of an experience base.

These four areas include 70% and 71% respectively of all answers and show an important conclusion:

People want to learn from others' experiences by reading their reports and by interacting with them – either formally in a study group or informally through socializing. Written procedures are not the answer.

	Developers		QA managers	
Vehicle	Number	Rank	Number	Rank
Study groups	5	3	4	2
Experience bases	10	1	5	1
Courses	3	5	1	7.5
Socializing	6	2	3	3.5
Discussions	1	9.5	1	7.5
Teambuilding	1	9.5	1	7.5
Individual studies	1	9.5	0	11
Reports, documentation	4	4	3	3.5
Job rotation	1	9.5	1	7.5
Multidisciplinary group	1	9.5	0	11
Working meetings	1	9.5	2	5
Include consultants	2	6	0	11

Table 3 Vehicles for knowledge transfer

Why they don't think SPI is a good idea

Those that don't do SPI

The companies that participated in the ESPINODE industrial survey and answered that they didn't do SPI were asked three important questions: Why don't you do SPI, what do you do to improve and what kind of help would you need to improve yourself? Since the questions were open-ended, we received quite a lot of different answers. We have selected categories, starting with the most popular one and then kept on including categories of decreasing popularity until we have reached at least 70% of all respondents. The majority of the answers ended up in one of a small number of categories, as shown in Table 4.

When we look at the results, there are two things that catch the eye: There is a strong agreement among the companies on why they do not need to do SPI and on what they do instead. Only two categories are needed in order to include 70% of all responders. The agreement is less when we come to the help needed. This is as expected – it is always more easy to diagnose a problem than to agree on a cure.

The fact that many companies claim that they do not have the necessary resources available indicates that they do not believe that SPI is a sound business proposition. When they say that they do not have enough resources to do SPI, they are really saying that they do not believe that they will make money out of it. They lack a cost / benefit analyses that will show how SPI will help their business. The problem may partly be connected to the way QA has been introduced. QA was supposed to increase efficiency and when the results did not materialize, this also contaminated the idea of SPI which at least partly has been sold as a QA technique.

Question	Category	%
Why don't you do SPI?	Do not have the resources available	40%.
	SPI costs too much	30%
What do you do to improve (instead of SPI)?	Improve and increase our know-how	52%
	Introduce new tools and methods	29%
What kind of help do you need?	Consultancy	24%
	Networking	24%
	Formative help on the concept of SPI	18%
	Financial help	14%

Table 4 Those that don't do SPI

What do they need?

In order to get an understanding of the companies' needs, we asked them what they considered their most important challenge for the next two to three years. The responses are summed up in Table 5. By and large, all companies – whether they do SPI or not - agree on their most important challenges for the future. Note, however, that while the SPI companies talk about improving efficiency, the others talk about improving innovation. Thus, it seems that the companies that do SPI focus on process efficiency while the rest focus on being innovative. Many companies seem to believe that SPI is a barrier for innovation – or at least that it doesn't help. Being innovative is much more critical for small companies, which depends on innovation in order to keep their competitive edge.

In our opinion, the fear that SPI will destroy or hinder creativity stems mainly from bad experiences with QA that has degenerated to a rigid control regime plus an enormous amount of documents that nobody outside the QA department really needs.

To get out of this sordid state, we need to disconnect SPI from QA, at least until QA move from being a controlling bureaucracy to being a service to the development projects. In addition, we need to develop a way of basing SPI on a sound business cost / benefit analysis so that the companies see that SPI really help them make money.

	Most important challenge for the next years	%
Companies that do SPI	Improve development efficiency	30%.

	Increase customer satisfaction	24%
	Promote company growth	23%
Companies that do not do SPI	Increase customer satisfaction	29%
	Be more innovative	28%
	Promote company growth	19%

Table 5 Company needs

The SPC inheritance

The concept of SPI stems from production industry. Their concept is simple and straightforward to apply:

1. Get the process under control by identifying and removing sources of variation.
2. Run controlled experiments where one or a few process parameters are varied in a controlled manner and collect data.
3. Analyze the collected data in order to identify the best candidate for improvement and implement the necessary changes.
4. Collect data to verify that the changes have had the expected impact.
5. Repeat the process from step 1.

Several persons – at some point in time event including one of the authors of this paper – thought that this improvement paradigm could also be used for software. Some still believe so. Level 4 and 5 of CMM are almost pure SPC both in concept and attempts. Software developers by and large consider SPC to be close to irrelevant for software development and they are probably right. The most important reasons for this are that SPC assumes:

A technologically stable process that is repeated a large number of times - typically several hundred times. This will probably never be the case for a software development process.

That the variation factors can be brought under statistical control. Given that a large part of the variation in a software development process stems from the developers, this is a rather unrealistic assumption.

Thus, SPC is out. To the degree that SPI pushes the use of SPC or is in any way connected with SPC, it will score low with software developers, and probably also with software managers. QA managers, however, too often think that SPC is a great idea – also for software.

The problems - as seen from the manager

The main problem with SPI as seen from the manager's side is that it consumes resources. This was given as the main reason by 40% of all managers asked. SPI is seen as an activity that takes resources – people and money - away from development which often is too late already due to lack of people. Even if managers believe that SPI will enable them to run the project within time and budget in the end, the end is too far away to interest them. A possible solution would be to hire more people but there are at least two obstacles to this solution:

Good developers is a scarce commodity.

Hiring more people will cost money and the improvement promised by SPI will only materialize somewhere in the future. It is a general experience that a sure expenditure today will usually outweigh an unsure income somewhere in the future.

An extra problem – especially as seen from middle management - is that SPI can only succeed by involving the developers. This is seen as a threat to middle management since involving the developers mean to give them more power – power that usually is taken away from middle managers.

The problems - as seen form the developers

Since SPI is marketed as a part of QA, developers automatically associates it all the misguided attempts to implement QA as a document driven control regime. Both “control” and “document” are terms that score low on the developers’ scale of enthusiasm. They feel that they are hindered in being creative and innovative – which is what development is really all about.
In addition to all this, QA – and thus SPI – is seen as a source of a never-ending steam of procedures for this and routines for that in addition to an insatiable lust for more documents. The problem is not the documents, but the justified suspicion that these documents do not add to the product’s value – they are just there to satisfy somebody’s need for control and control translates into “I don’t trust you”.
A survey of working relation in the North Sea 0 illustrates this: “The Norwegians coursed the way they were forced to work. They were forced into what they felt were idiotic QA procedures. They were proud industrial workers who had to follow routines and a documentation regime that the on-shore industry had outgrown a long time ago. They had to wait for a working permit even for small jobs and the whole place was crowded with managers on all levels, with certificates and detailed working procedures”. In the offshore area this lead to eight years of continuos strikes and worker unrest. Software developers have not come that far – yet.

QA managers versus developers

One of the points that are driven home to us quite often is that the world looks quite different depending on whether you see SPI from the QA manager or from the developers’ side. Some times one wonders: Do they really work for the same company or do they just share a mail address?
Some times the differences are important – for instance that developers and QA managers agreed that developer involvement was important for the success of SPI, while they did not agree on what developer involvement really meant. For developers and development managers, developer participation consisted of the following factors:

A strong influence on how their work is done.

Participation in the improvement work by making improvement proposals.

Participation in the development of procedures for development work

Participation when the company sets improvement goals.

Responsibility for part of the improvement work.

Being regularly informed of the status on the company’s improvement work

Discuss the principles for SPI regularly discussed with the management.

However, when we asked the QA managers, they produced the following, rather short list of participation factors:

Participation in the development of procedures for development work

Responsibility for part of the improvement work.

What is most revealing in the factors identified by the QA mangers is what is ***not*** included. These are factors such as participation by making improvement proposals, by setting improvement goals, and by being informed of the status in the improvement work. It is as if the QA managers say that SPI is the responsibility of the QA department. The developers should stand by and watch.

This problem was highlighted when we checked the two groups' position regarding feedback sessions. It turned out that the QA managers considered feedback sessions to be sessions where the developers were allowed to see the data. The development managers, on the other hand, considered feedback session to be a part of data collection and analyses – which make sense. Our experience is that the developers' participation in feedback session is absolutely essential for data analyses and interpretation.

Where do we go from here – out of the quagmire

Even though the situation in many ways is rather bad, there are hopes for the future. Firstly, there are companies that have succeeded in their SPI work. These companies have been able to make the QA department and the software development department co-operate in a way that contributes to increased quality for the customer and increased efficiency for the company. In these companies the developers are involved in the improvement work and the management supports it. Secondly, the SPI field has matured over the years, and we have today a better understanding of how to implement SPI so that it caters to the needs of software development and the needs of each specific company.

The problems discussed in the first part of this paper are diverse. This diversity could have resulted in a large set of solutions. We think, however, that there are a few areas that dominate, and we will therefore focus on solutions to these most important obstacles to SPI.

SPI and QA

W.S. Humprey introduced in 0 what he called Software Engineering Process Groups (SEPG's) and these groups should among other things, take care of process improvement activities. In the discussions of why these groups were needed; why not let the QA department take this job, he stated ”At least from a parochial development viewpoint, they (QA) are thus often viewed as the enemy, or at least as an unnecessary annoyance". When we first read the book, we thought this was an American problem and not valid for the European way of working. Unfortunately, we were wrong. Our data and later experience has shown that this is a relevant statement, at least for part of the European software industry. The SPI community therefore has two alternatives:

Either: Keep away from the QA department as best they can.
Or: Be an integrated part of the QA activities.

The first alternative may work for some time, but is not a solution for the future. The second alternative is therefore our choice. The main reason for this is that SPI and QA overlap. QA shall verify that the employees have applied their expertise properly, ISO 9000 requires continuous improvement, etc. SPI is about learning to work smarter - changing the process to get an environment in which people can do a better job. The obstacles to SPI are thus not the tasks to be done or a lack of common interests. The problems stem from the implementation. There are two main areas where today's implementation has led to conflicts of interest between QA and SPI.

Organisation

QA has traditionally been organised as a separate department - at least in large companies. This organisation has often resulted in little contact and co-operation between QA and the developers. SPI, on the other side, depends on close co-operation with the development department and the developers – preferably on a daily basis. SPI can not succeed without active participation from the employees.

Control versus help

The separation of QA from the developers can lead to a situation where QA become a part of the management - controlling the developers instead of helping them. SPI is about helping people to do a better job.

The solution to these problems is simple - at least in principle. Establish a common platform for SPI and QA where they can work together toward a common goal - looking for new improvement opportunities. In this setting, QA activities must be in-project activities and QA people should take part in the daily work of the development projects. This is the only way that QA people can get hands-on experience and thus learn about the developers' needs - problems to be solved and opportunities to grab.

Make SPI and QA unify their effort, looking for improvement opportunities.

SPI and software development

In order to succeed in SPI, we have to understand and accept the nature of software development. We have to understand that software development is a creative and innovative process. In spite of all the methods and tools appearing on the marked at a high frequency, software development is in the end a people process.

Another aspect of software development is the technology focus and rapid technology changes. These changes may be in conflict with one of the main fundaments of SPI - that it should be fact based. Facts come from measurements, and the question is how we can collect enough data to draw conclusions based on statistical data analysis in an ever-changing environment. The answer is simple - we can not. That does not mean that we have to give up. The solution is to use the data as indicators and include the experts - the people related to the measured process - and their interpretation of the data. This implies that you have to take risks. You can not be sure that the interpretations are correct. A few data combined with expert knowledge are, however, far better than expert knowledge alone. The measured facts are there to aid the experts in reaching their conclusions and keeping this process on the right track. The risks that rise from this way of reaching conclusions have to be controlled. Thus, risk analyses has to be an integrated part of any SPI program.

SPI should be based on a combination of measurement data and expert knowledge - controlled by a risk analyses.

SPI and the developers

Software development is one thing, software developers is quite another. How can we motivate them to get involved with SPI? Rigid procedures and control are definitely not the answer. This does not mean that developers do not want procedures at all or that all project tracking is considered to be control. Firstly, lack of procedures is often considered a problem by the developers, and definition of new procedures can be the result of SPI activities. Procedures will reduce the time spent thinking about small problems - giving more time to be used on creative tasks 0. On the other hand, procedures are not considered to be an aid for knowledge transfer (see Table 1). The problem with existing procedures is, however, that they not always reflect the needs of the developers. Secondly, project tracking means measurements, and measurement data can be collected as long as they are considered useful by the developers and not used as a vehicle for control. The solution to both these problems is developer participation. Our experience is clear; if the developers are involved, they will be motivated. In SPI, involvement means to participate in:

Defining the improvement strategy - problems to be solved and opportunities to take advantage of

Define the metrics and the data collection procedures

Interpretation of the measurement data

Identifying and implementing improvement activities, i.e. defining procedures.

SPI can not solve the developers’ problems without their participation

SPI and the business

To make it easier to sell SPI to the management, we should be able to document the cost and benefit of SPI. SPI methods have been used to document the benefit of inspections, the effect of new tools or methods etc. 0, 0. Little have, however, been done up till now in order to document the effect of the SPI itself. Some data has been published 0 - all of them positive, but not all of them widely accepted.

The SPI community is fact based and they thus require evidence in order to institutionalize the results of improvement activities. Why do they do not put the same requirement on themselves - perform cost benefit analyses of the total company SPI activity and document the results. This is a challenge that the whole SPI community has to meet in the near future if they want a wider acceptance and use.

Cost benefit analysis of SPI have to be put forward in order to sell SPI to managers

Conclusions

SPI is about learning to work smarter, and thus everybody in the software community should be interested. So, why do they not believe that SPI can help them? First, the discipline of software engineering has matured over the years, but is still to some degree based on hero programmers that believe that SPI will be like QA - more paper work and control. Second, SPI means investments and managers do not like to invest without at least some kind of documentation of the benefits.

Tomorrow's organizations on the other hand, will be organizations that are continuously looking for improvement opportunities. The improvements should be based on a few vital elements:

Facts - measurements combined with knowledge - in order to solve the right problems. Uncertainty should be handles through risk management.

Participation - in order to motivate the developers and keep the improvement program alive.

Learning - both on the individual and organizational level.

Minimal paper work and overhead - keeping creativity and innovation alive.

Cost/benefit analyses in order to defend the investments.

References

Dybå, T., Important Success Factors in SPI, Ph.D. theses (to appear in year 2000)

Carlsen, J.E., Fosnæss, M., Process Improvement Survey, NTNU 1999 (in Norwegian)

Gilb, T., Graham, D., Software Inspection, Addison Wesley, 1993, ISBN 0-201-63181-4

Herbsleb, J et al, Benefits of CMM-Based Software Process Improvement: Initial Results, in: CMU/SEI-94-TR-13

Humprey, W. S., Managing the Software Process, The SEI Series in Software Engineering, 1989, ISBN 0-201-18095-2

McGibbon, T., A Business Case for Software Process Improvement, in: A DACS State-of-the-Art Report, Rome Laboratory, 30 September 1996

Paulk, M.C., Weber, C.V., Curtis, B. and Chrissis, M.B. The Capability Maturity Model, Addison-Wesley, 1995

Røyrvik, E.: Cowboys and rebels in the North Sea, in: Gemini - Research news from SINTEF and NTNU, No. 3 - June 1999 (in Norwegian)

Stålhane, T., ESPINODE Industrial Surway, in: QIS Newsletter, No. 4, May 1999.

Uchimaru, K., Okamoto, S., Kurahara, B., TQM for Technical Groups, Productivity Press, Portland, Oregon, ISBN 1-56327-005-6

SINTEF is an independent, not-for-profit research foundation based in Trondheim and Oslo, Norway. Our role is to encourage innovation and improve competitiveness in Norwegian industry and public administration. In doing so, we maintain close links with the technical Universities in Trondheim and Oslo, collaborating on projects, and sharing equipment and other resources.
SINTEF is not a publicly funded organization. A very small part (less than 4%) of our income is from a public grant; most of our operating revenues arise from contract research and development work carried out for industry and the public sector in Norway and elsewhere.
With over 1800 employees and a turnover of NOK 1.4 billion, SINTEF is Scandinavia's largest independent research organization. It is organized into eight separate research institutes, covering all major scientific areas and industrial sectors. Refer to our web site www.sintef.no for further information.

SINTEF has over the years been a leading company in the area of software engineering and have broad and deep experience in this area. This experience serves as a sound basis for our Software Process Improvement (SPI) work. Our major SPI activities are:

The SPIQ program – Software Process Improvement for better Quality. This is a national Norwegian program that is partly a national ESSI type project and partly a co-operation with Norwegian industry to increase the use of process improvement methods in Norwegian software industry.

The QIS project – Quality Improvement in Scandinavia. This is an EU sponsored project that shall give help and assistance to PIEs in Norway and Sweden and market the concept of SPI to software industry in these two countries.

Tor Stålhane was born in 1944 and became a M.Sc. at the Norwegian Institute of Technology, University of Trondheim (NTNU) in 1969. During 1969 to 1985 he worked at SINTEF - RUNIT, department for languages and compilers. From 1985 he worked on his Ph.D. studies and finished his thesis on software reliability in 1988. From 1988 he was back at SINTEF where he mainly worked with quality assurance and software safety and reliability. In 1997 he became professor in Computer Science at the Stavanger Polytechnic. During the latest decade he has been mainly been working with safety analyses of software intensive systems and measurement based process improvement.

Kari Juul Wedde was born in 1951 and became a B.Sc. at the Technical College of Trondheim in 1972. After that she has continuously updated herself by following graduate and postgraduate courses at NTNU. In 1973 she started to work for SINTEF. The work at SINTEF has given her a long and broad experience in software engineering, ranging from compiler construction to telecom applications. Her main areas of expertise during the last years has been software testing, quality assurance, measurement based software process improvement and safety analyses of software intensive systems, and she has several international publications in these areas.

Software Measurement Frameworks to Assess the Value of Process Improvement

Dr. Nancy Eickelmann
NASA IV&V Facility
Software Research Laboratory
100 University Drive
Fairmont, West Virginia 26554
+1 304 367 8444
http://research.ivv.nasa.gov/~ike
Nancy.Eickelmann@ivv.nasa.gov

Introduction

Business process improvement (BPI) has been used in organizations for several decades promising great financial rewards and often delivering on those promises. Process improvement approaches have included Total Quality Management TQM, the Software Engineering Institute's Capability Maturity Model CMM, and more recently ISO-9000. All of these efforts share a customer focus towards measurable business process improvements that promise cost reductions and cycle time improvements. Unfortunately, the measurement frameworks used to deploy and manage business process improvement initiatives are frequently not linked to the organization's high-level strategic goals. Thus, organizations might achieve international recognition for the quality improvements in their products while simultaneously failing to meet their strategic goals and objectives.

The Balanced Score Card (BSC) Framework provides the necessary structure to evaluate quantitative and qualitative information with respect to the organization's strategic vision and goals. NASA is organized into 12 regional centers each with its own strategic plan that documents their respective vision, mission, and goals. The BSC is applied to the IV&V center in this study. The goals and objectives of the other 11 centers will be discussed in future work and integrated into an overall NASA BSC. The Enterprise goals are common to all NASA technology enterprises of Space and Earth Science, Human Exploration and Development of Space Enterprise, and Aero-Space Technology Enterprise. The goals are mission success, safety of people and property, and cost containment.

There are two categories of measures used in the BSC the leading indicators or performance drivers and the lagging indicators or outcome measures. The performance drivers enable the organization to achieve short-term operational improvements while the outcome measures provide objective evidence of whether *strategic objectives* are achieved. The two measures must be used in conjunction with one another to link measurement throughout the organization thus giving visibility into the organizations progress in achieving strategic goals through process improvement. The development of a core set of metrics for implementing the Balanced Score Card is the most difficult aspect of the approach. Developing metrics that create the necessary linkages of the operational directives with the strategic mission prove to be fundamentally difficult as it is typical to view organizational performance in terms of outcomes or results rather than focus on metrics that address performance drivers that provide feedback concerning day-to-day organizational progress.

NASA IV&V applies a sophisticated level of understanding concerning the application and measurement of process improvement in an organization and its strategic role in success or failure of the overall mission. Specific measures are used to track and evaluate the progress towards technological support for consolidation, modernization, interoperability, standardization, and increased use of commercial off the shelf COTS products, high customer satisfaction, training goals and high return on investment ROI for the agency. These measures represent linkages from key processes that are required to achieve organizational goals and objectives. An analysis of NASA IV&V key processes is facilitated by the introduction of the Test Technology Evaluation Framework (TTEF). The TTEF is applied to measure and evaluate software and system test technologies. Thus, providing the necessary structure to measure key processes central to IV&V practice. Specific contributions of this work include:

- a characterization of a core metrics set required for a BSC approach in a software development environment applying IV&V technologies
- a characterization of key process improvement measurement requirements focusing on the IV&V process
- an identification of open measurement issues and further research

This paper identifies the critical linkages between financial measures of past performance and key indicators (measures) of future performance based on process measurement and improvement. The paper is organized by named sections starting with an introduction of the paper, a description of the Balanced Score Card as developed for IV&V, a description of the Test Technology Evaluation Framework TTEF and a summary of measured results from applying the TTEF and lessons learned.

Overview of the BSC

The BSC architecture was intended to provide a framework for industry and for-profit organizations. The framework facilitates translating the strategic plan into concrete operational terms that can be communicated throughout the organization and measured to evaluate its day-to-day viability. The three principles of building a balanced scorecard that is linked through a measurement framework to the organizational strategy include:

(1) defining the cause and effect relationships,
(2) defining the outcomes and performance drivers,
(3) linking the scorecard to the financial outcome measures

Table 1.1 Balanced Score Card Government Vision and Strategy Mapping to Operational Focus – Source [2]

	Government
Financial	**“How can we reduce costs and not compromise our mission?”**
Customer	**“To achieve our vision how do we want our customers to perceive us?”**
Internal Business Process	**“To satisfy our customers what business processes do we need to excel at to differentiate us and create a COE?”**
Learning & Growth	**“What infrastructure do we need to sustain our ability to change and improve?”**

The initial steps of BSC engage in the construction of a set of hypotheses concerning cause and effect relationships among objectives for all four perspectives of the balanced score card. The measurement system makes these relationships explicit. Therefore, they can be used to assess and evaluate the validity of the BSC hypotheses. The questions asked in each category of the four perspectives provide a segue into the cause effect diagramming activity see table 1.1.

This paper incorporates findings from prior case studies in [2] that have been used to identify key factors that differentiate the use of the BSC for government or non-profit organizations versus industry or for profit organizations. The strategic goals for the NASA IV&V Facility are discussed in the context of core processes related to software verification and validation technologies. The core metrics set that was developed for the BSC is discussed in each section respective to its area of focus as linked through the cause-effect graphing topology. The next section provides the NASA IV&V Facility strategic vision and goals that are used to direct the BSC effort.

NASA IV&V Strategic Vision and Goals

The strategic plan contains the vision, goals, mission and values for the organization. The Government Performance and Results Act, GPRA requires all federal agencies to establish strategic plans and measure their performance in achieving their missions. The vision and goals are stated below. NASA IV&V strategic vision and goals statements as presented in the 1999 Strategic Plan.

> Vision: To be world-class creators and facilitators of innovative, intelligent, high performance, reliable informational technologies that enable NASA missions.
>
> Goals: To become an international leading force in the field of software engineering for improving safety, quality, reliability, and cost performance of software systems; and to become a national Center of Excellence (COE) in systems and software independent verification and validation.

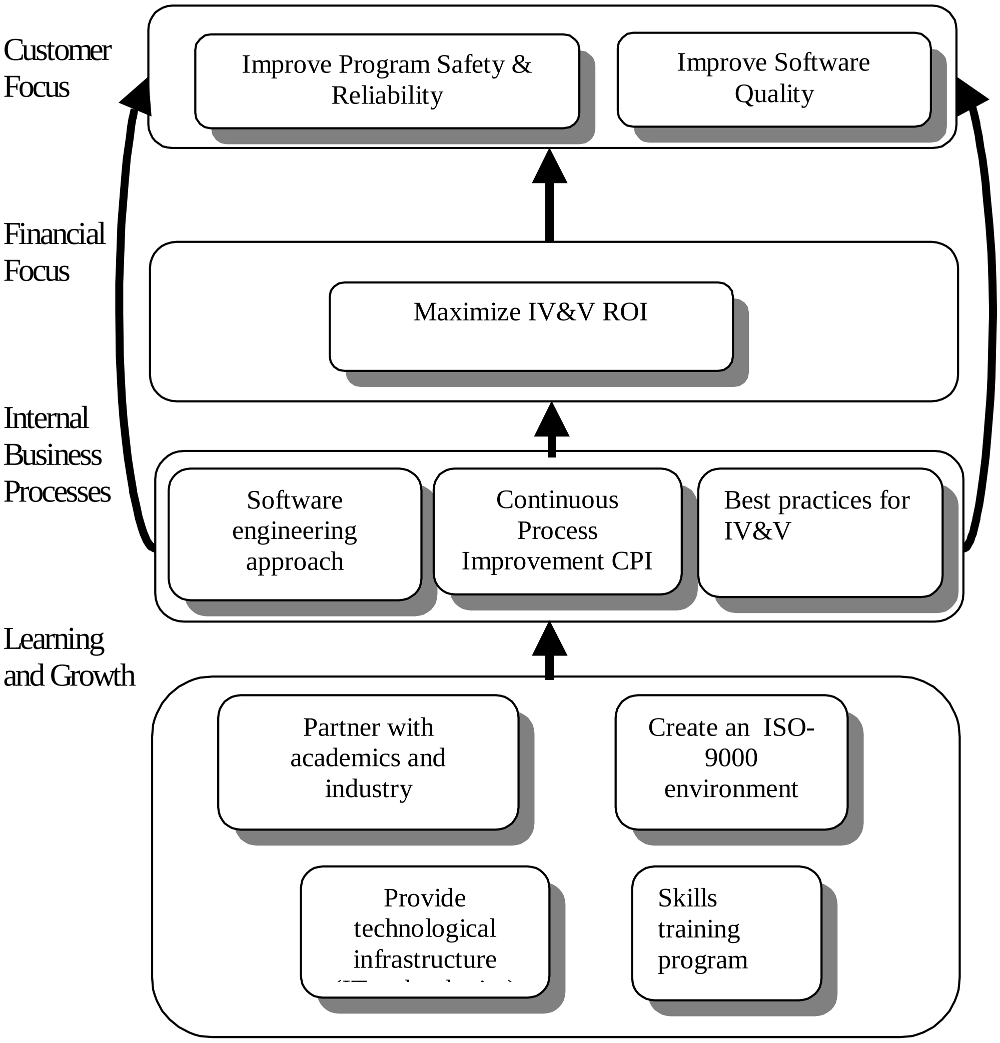
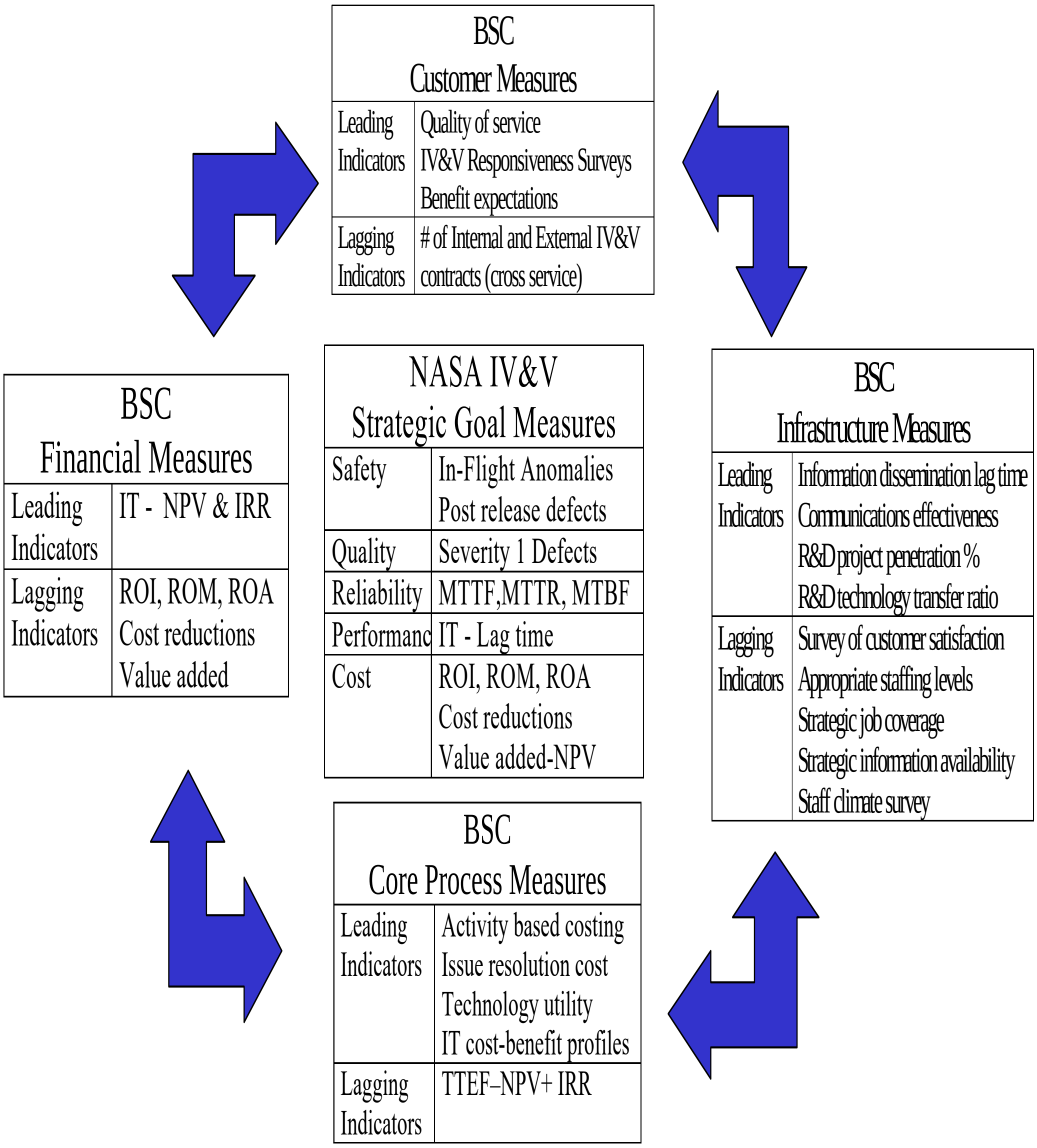

Customer Focus
Improve Program Safety & Reliability
Improve Software Quality
Financial Focus
Maximize IV&V ROI
Internal Business Processes
Software engineering approach
Continuous Process Improvement CPI
Best practices for IV&V
Learning and Growth
Partner with academics and industry
Create an ISO-9000 environment
Provide technological infrastructure
Skills training program

Fig. IKE. 1.1 : BSC objectives hierarchy.

The objectives for each tier of the BSC are shown in Fig. IKE.1.1. The figure also depicts the mutation of the BSC geography as proposed by Kaplan to place the customer focus at the top level for government and not-for-profit organizations. The BSC objectives hierarchy is used to generate the necessary metrics to measure strategic as well as tactical progress for all four tiers. The metrics developed to track mission success relative to the customer focus of improved safety, reliability and quality and the financial focus of reduced costs at maximum benefit of technology are under review. The initial core set of metrics for leading and lagging indicators is shown in Fig IKE.1.2.



BSC
Customer Measures

Leading Indicators	Quality of service IV&V Responsiveness Surveys Benefit expectations
Lagging Indicators	# of Internal and External IV&V contracts (cross service)

BSC
Financial Measures

Leading Indicators	IT - NPV & IRR
Lagging Indicators	ROI, ROM, ROA Cost reductions Value added

NASA IV&V
Strategic Goal Measures

Safety	In-Flight Anomalies Post release defects
Quality	Severity 1 Defects
Reliability	MTTF,MTTR, MTBF
Performanc	IT - Lag time
Cost	ROI, ROM, ROA Cost reductions Value added-NPV

BSC
Infrastructure Measures

Leading Indicators	Information dissemination lag time Communications effectiveness R&D project penetration % R&D technology transfer ratio
Lagging Indicators	Survey of customer satisfaction Appropriate staffing levels Strategic job coverage Strategic information availability Staff climate survey

BSC
Core Process Measures

Leading Indicators	Activity based costing Issue resolution cost Technology utility IT cost-benefit profiles
Lagging Indicators	TTEF–NPV+ IRR

Fig. IKE.1.2 : Balance Score Card core metrics set

Customer Measures of Mission Success

The customer focused objectives of improved mission safety, improved mission systems and software reliability, improved systems and software quality, and reduced costs each have specific measures and targets that are used to evaluate whether or not strategic goals are achieved and to what degree. This requires identifying viable measurement strategies for software IV&V in the NASA context. A difficulty in measuring thematic aspects of the customer focus arises from the co-variation that exists among the themes. To illustrate this point we must first define what we mean by safety, reliability and quality in customer themes and then define the relationships among these objectives.

➢ Safety is defined as freedom from accidents or losses. This is an absolute statement, safety is more practically viewed as a continuum from no accidents or losses to acceptable levels of risk of loss.

- Reliability is defined in terms of the probabilistic or statistical behavior, that is the probability that the software will operate as expected over a specified period of time.
- Quality is defined in terms of correctness and number of defects. Correctness is an absolute quality, it is also a mathematical property that establishes the equivalence between the software and its specification.
- Cost is more complex than it appears, direct or absorption costing may be applied and alters what costs are included and therefore what costs may be reduced. The focus of the paper does not rely on the differences inherent to these two approaches and therefore defers discussion of this topic.

The relationships among these customer themes are significant as they are not independent of one another and therefore must be analyzed based on their degree of covariance and interaction. The relationships are diagrammed in Fig. IKE 1.3 and depict the current accepted understanding. Safety requires that unsafe states cannot be entered from any point of function of the system. It is possible for the systems to function reliably that is without failure and still enter unsafe states of operation. A system can be completely correct and defect free and still enter unsafe states. There are many documented examples of these properties in the literature and many devoted specifically to documenting the complexity of software safety issues. The safety of a system is a result of its safe operation in a specific context or environment.

The NASA IV&V facility must document the increase in software and systems safety, reliability and quality that are attributable to IV&V technologies. This requires that the contribution that is made towards meeting required targets through the application of IV&V activities must be quantified. This requires that each aspect be evaluated relative to some objective target. The value add of IV&V is measured as the sum of overall reduction of distance from the target values (see Fig. IKE 1.4). This provides a measure of overall impact to mission success. The relative reduction of “Euclidean Distance” from the shuttle safety target of no losses, attributable to IV&V efforts specifically, is documented and integrated into the overall model that sums the total reduction of distance from the three targets of safety, reliability and quality, relative to a fixed cost. There are many measures that can be collected to evaluate the added value of IV&V for software and system safety; this is only one approach.

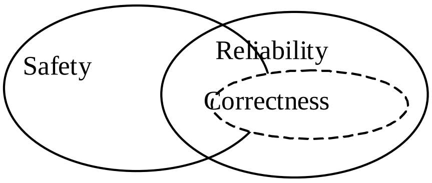


Fig. IKE 1.3 relationships among customer themes of mission success through safety, reliability, and quality at reduced costs.

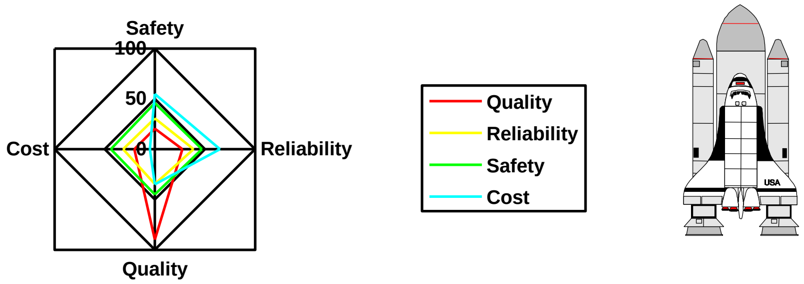


Fig. IKE 1.4 The customer theme of mission success through safety, reliability, quality, at minimum cost is shown in the graph depicting the interdependence of the themes.

The measurement of the contribution of IV&V in improving safety, reliability and quality while reducing cost is discussed in the following sections. The contribution of IV&V to shuttle safety is difficult to measure directly. It is therefore necessary to make assumptions concerning those factors that would impact safety and to what degree. It is assumed that a reduction in the probability of failure is a contribution to increased safety. A reduction of the number of In Flight Anomalies IFAs of a severe nature due to IV&V identification and removal is a contribution. An independent evaluation of potential failure modes that results in identifying previously unidentified hazards is a contribution. The elimination or mitigation of hazardous states or their potential is quantified relative to probabilistic measures of hazard occurrence and the likely severity.

The contribution of IV&V to shuttle reliability is more directly attributable to the specific verification activities that are applied during the Shuttle software development process towards defect management. Research investigating the ramifications of testing strategies for reliability provides quantification of benefits relative to specific IV&V activities. A minimization of estimated residual faults is provided according to the sequence of testing strategies and the duration of those test executions. For example the number of defects detected by applying functional, decision, data flow and mutation test methods in sequence. The CPU execution time or the number of test cases can measure test effort. As the test effort increases defects detected can be optimized through applying more optimistic or pessimistic test strategies. The resulting increase in reliability is measured by increased MTTF or improved failure intensity profiles and is quantified as a reduction in the distance from the reliability targets of subsystems undergoing IV&V.

The contribution of IV&V to shuttle quality is measured as a reduction of defect density trends through process improvement paradigms such as traversing the CMM stages from levels 2, 3, 4 to level 5. The intuition behind this model is that the measurable impact of process improvement is in the reduction of the cost of rework. In addition, the rework cost avoidance of detecting defects of severity 1; severity 2 and severity 3 can be quantified relative to phase of detection and level of severity. The reduction of defect density is measured as a reduction of distance from the overall quality objective measured in defect density according to severity.

The shuttle software safety, reliability and quality are measured according to the BSC core metrics of In Flight Anomalies IFAs, post release defects, severity 1 & 2 defects, mean time to failure MTTF, mean time to recovery MTTR, and mean time between failures MTBF. A key software engineering practice that is in part responsible for the software's high degree of assurance is that of reuse. The shuttle program has a sophisticated approach to the sustaining of the core functionality of the systems and software while providing controlled evolution to support new missions. The measurement of reusability of key architectural, design, code and test artifacts is currently under study as part of the BSC research.

Financial Measures

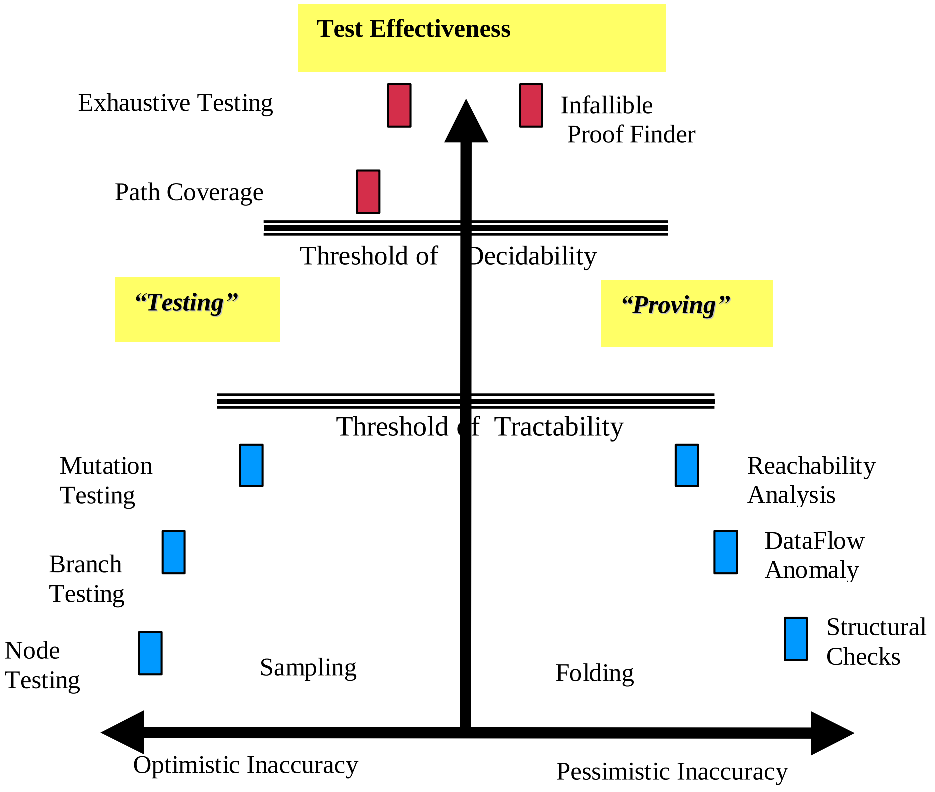
The fundamental ROI model is the ratio of net income to total investment. This is a financial measure and has been used predominantly to measure a manufacturing firm's efficiency in allocating resources. In the financial ROI model the numerator is net income for a project or time period and the denominator is some measure of total investment respective to the project cost or capital expenditures for the time period. The measurement of IV&V benefits must measure improved safety, reliability, as well as improved quality.

Improved quality is typically measured as a benefit of process improvement and indirectly attributed to an overall reduction in rework. Specific examples of applying the rework avoidance concept to ROI are documented in the literature and state substantial savings associated with rework avoidance. Raytheon Systems Corporation reported cost savings of $15.8 million for 15 projects over a four-year period. Raytheon documents an ROI of 7:1 based on $4.48 million return for $580,000 invested. Hughes Aircraft reported cost savings of $9.2 million over a three-year period. Hughes documents an ROI of 4.5:1 based on $2 million return on $400,000 invested. The Aircraft Software Division at Tinker Air Force Base reported an ROI of 6.35:1 based on a return of $2.9 million for $462,100 invested.

Improvements to safety are typically quantified by obviating some hazardous state and measuring the resulting reduction of risk. Improved reliability is measured as a reduction in the intensity failure or improvement in mean time to failure measure. All of these benefits are harvested in a specific context of development. In the case of the shuttle a product line reuse process is applied to allow the core shuttle software to be systematically evolved over time and adapted for new mission requirements in a timely manner. The reuse paradigm also provides a unique context to facilitate the V&V of the software with each new version.

Process Measures

A primary business process of IV&V is applying test technologies. Test technologies may be applied to improve safety, reliability or quality at a minimum cost. The "Euclidean Distancing" method used for the customer perspective is supplied with metrics from the TTEF evaluation of test technologies. The evaluative framework is



Adapted Young & Taylor, ACM 1989

Fig. IKE.1.5 : Software test hierarchy

constructed to support an accurate cost and benefit analysis of the application of emerging test technologies. The efficacy of the framework's value is demonstrated by applying it to evaluate the adoption of specification-based test technologies. Specific examples are chosen that use commercially available test tools and object-oriented modeling and specification documentation tools. A comparison is conducted between current practice for evaluating technologies and our Test Technology Evaluation Framework (TTEF). The framework provides a rigorous and repeatable methodology that guides the test manager in the correct usage of measurement and evaluation models. The Test Technology Evaluation Framework takes a comprehensive view of the critical factors of:

- *test effectiveness* based on a comparison of test detection rates and test failure rates
- *test productivity* based on increased production of verifiable test sets
- *test schedule compression* based on full life cycle analysis.

The TTEF evaluates the degree of achievement of the primary goals including test productivity, test effectiveness, and test schedule compression. The TTEF uses a comparative analysis and is designed to capture the principle benefits associated with a technology as well as the complete cost structure associated with a technology.

Test productivity must reflect the success of the tests as well as the quantity of test cases executed per period of time. Time may be execution time or calendar time, as measured in mean time to test completion (MTTC). Test Pass/Fail rates is used to evaluate test oracle effectiveness. A significant contributor to determining test Pass/Fail rates is the test selection criteria that are used. For instance, test selection from the input domain may result in different points of the input domain to be labeled as "ϕ", "σ" for (failure and success). The probability distribution Q over the input domain D allocates a probability of selection of tests *t*, the sum of which is equal to 1. In random test selection "ϕ", "σ" will appear in even proportions (for large test sets). When using selective test criteria testers seek all *t* labeled "ϕ", and seek to select no *t* labeled "σ". An indicator variable δ is used to represent all points in the input domain that are labeled either "ϕ", "σ". To evaluate test effectiveness test strategies may be compared with respect to their detection rates for debug testing or compared to their failure rates for operational testing.

To evaluate potential test effectiveness of test technologies they may be categorized according to their effectiveness by mapping them to a hierarchical taxonomy such as proposed by Taylor and Young (this is only one such taxonomy there are several that might be applied). The Taylor and Young taxonomy categorizes test strategies according to the test methods sensitivity to revealing faults for a given level of effort. The primary classification boundary is whether the test method is known to provide optimistic or pessimistic inaccuracy in the test results. Optimistic inaccuracy is typically test methods that apply sampling techniques such as node testing, branch or statement testing, and mutation testing. Pessimistic inaccuracy is typically a result of applying folding techniques such as data flow analysis, reachability analysis, and theorem proving techniques. The figure of Test Effectiveness (see Fig. IKE. 1.5) is adapted from Taylor and Young's taxonomic diagram and is altered in a single aspect. The upward vector is labeled as effort in the their taxonomy and was relabeled as test effectiveness based on our findings. Test effort is typically measured either in number of hours spent by the tester or clock hours of the test execution. Neither measure captures significant components of test effort such as oracle verification time (manual versus automated) or test case selection for non-random test case generation methods. Degrees of inaccuracy however are directly related to the degree of test effectiveness in revealing faults. This differentiation allows us to measure effectiveness relative to the test strategy and productivity relative to the overall test process.

The benefits of schedule compression must be viewed from a lifecycle perspective as test strategies and technologies may provide a compression of test time for one aspect of the test process and simultaneously extend overall test time by increasing some other aspect of the process. For example, automating test case generation can compress test selection time while extending the time to conduct test pass/fail evaluation with a manual test oracle verification procedure.

The software reliability research community has developed models to estimate the MTTC based on the number of test cases that are executed. These reliability models can be used to provide an estimate of test time required for a given level of confidence. The completion of test execution is based on achieving a desired level of accuracy or confidence in the state of the product. Thus the TTEF facilitates the efficient allocation of testing resources with measurable reliability impact analysis.

The Comparative Analysis

The efficacy of the framework's value is demonstrated by applying it to evaluate the adoption of specification-based test technologies. Specific examples are chosen that use commercially available test tools and object-oriented modeling and specification documentation tools. A comparison is conducted between current practice for evaluating technologies and our Test Technology Evaluation Framework (TTEF). The framework provides a rigorous and repeatable methodology that guides the test manager in the correct usage of measurement and evaluation models. The evaluation of the formulation of ROI of automated object-oriented test technology is conducted using traditional methods (such as applying industry benchmark's as published by Capers Jones). The shortcomings of improperly accounting for the fundamental process in when quantifying benefits of technology are demonstrated by comparison of methods. The Test Technology Evaluation Framework is also applied to the automated object-oriented test technology. The results of the comparison identify the common pitfalls of incorrect usage of ROI models and methods. It is shown that the underlying test process imposes restrictions of how the model of technological benefits is formulated and how it can be interpreted. The results demonstrate the improved reliability of ROI measures when applied correctly.

Naïve Analysis

The study begins with an evaluation of applying OMT technologies that enable automated specification-based testing technologies. The object modeling tool applied was StP/OMT (Integrated Development Environments, IDE). This is a development tool based on James Rumbaugh's object modeling technology and was used to develop the specification. The test planning and test case generation tool applied was StP/T (Integrated Development Environments, IDE). This is a test case generation tool based that generates test cases form the OMT specifications. The test execution was performed using XRUNNER. This is an automated software test tool. The cost savings are based on a cost avoidance of allocating a test engineer to develop test cases, an effort estimated to be one-person month. This cost savings amount was based on a published industry average of test case productivity (Capers Jones) of 20 to 300 test cases per month. The cost of adopting and using the technology was restricted to the 24-minute development time for the design specification of the OMT models that are subsequently used to generate the test cases. A COCOMO Man Month is equal to 152 hours. If only the model development time of 24 minutes is applied to test cost, the test benefit with automated OMT would be 151.5 hours of labor effort saved. The study reported that using OMT strategies and tools resulted in an ROI of 304:1.

TTEF Analysis

The TTEF analysis incorporates a lifecycle software test process perspective. The TTEF is designed to properly scope costs and benefits relative to the underlying processes that are the context for use of any development or test tool. Thus the TTEF provides a realistic analysis of expected outcomes of applying test technologies. The focus of the framework is to measure test productivity and test effectiveness relative to schedule compression or time savings. This requires that a full life cycle view of the process be applied to determine that gains made with respect to one aspect of the process have not negatively impacted another aspect of the process. This section will apply the TTEF to evaluate ROI and NPV for the OMT specification based test technologies.

The costs associated with adopting automated test technology based on OMT models are listed below. The costs would be similar for most automated testing tools that are associated with the paradigm shift to OO technology and rely on anticipated reuse benefits with respect to the technology window:

- Equipment purchase StP/OMT, StP/T, Xrunner*
- Reuse libraries*
- Training of personnel*
- OMT Specification development and analysis
- Verification of OMT model fidelity
- Maintenance and archiving of reusable assets
- Software updates

The asterisk marks those costs that would be considered initial costs in adopting test automation technologies. The benefits associated with adopting automated test technology based on OMT models are listed below. The benefits would be similar for most automated testing tools that are associated with the paradigm shift to OO technology and rely on anticipated reuse benefits with respect to the technology window:

- Test personnel time savings
- Automated test case generation and execution
- Test schedule savings
- Design model reuse*
- Test specification reuse*

The asterisk marks those benefits that may accrue to future projects based on reuse savings. The ROI = 0.1517: 1 as calculated by applying the above costs and benefits for the technology. This is significantly less than an ROI 304:1 as calculated under the naïve analysis.

NPV versus Rate of Return

A more discriminating approach is to apply net present value NPV and internal rate of return IRR. NPV evaluates the additional value provided the organization, IRR evaluates the rate of return for the project. NPV is typically of greater significance in the project decision. For example if one project earns 50% on a $1 million investment (IRR=50%, NPV=$500,000) and another earns 200% on $.10 investment (IRR=200%, NPV=20 cents), the dollar value in absolute terms is much more significant. The threshold for acceptance of a project is any NPV greater than zero. NPV and IRR with multiple independent projects will always lead to the same accept or reject decisions if NPV is positive IRR is less than the cost of capital. If the full costs and benefits of automated OMT technologies are applied an NPV value calculated using TTEF is NPV = $1819.00 (in year one with reuse over 10 projects). Assumptions in this calculation include: automated specification based testing initial cost estimate of $20,036.00 per seat, annual maintenance costs of $ 3,680.00, additional savings through reuse strategies of $2772.00 per additional program (this example uses 10 programs to calculate the extent of reuse), and uses a discount rate of 10%. The NPV calculation is well-suited to incorporating product line reuse benefits into the financial model.

Summary

Results of our case studies demonstrate that the rapid changes in software development, such as object-oriented methodologies, will require new approaches for measuring and evaluating software test technologies. The value of OMT technologies was significantly overstated when the scope of the cost-benefit analysis was underspecified, ROI 304:1. Conversely, the value of OMT technologies was understated when the scope was applied correctly and the underlying process characteristics, in this instance a specification based reuse process, was not taken into account, ROI 0.1517:1. Finally when both the scope of the cost-benefit analysis and the underlying process and reuse paradigm are incorporated into the analysis meaningful quantification of the value of the technology results.

Our traditional means of evaluating costs and benefits do not capture the essential characteristics of emerging test technologies based on reuse, specification-based test oracles, automated test case generation, and automated impact analysis to name a few. There were significant costs not accounted for in the naïve analysis including the cost of shifting to an object-oriented development process that combines design specification and test case design. The OMT model development and the subsequent evaluation of model fidelity are not considered. The result is cost shifting to tools, training, updates and required reuse of models. The degree of test effectiveness was not evaluated but was tacitly assumed as superior due to the virtues of automation. This is not a prudent assumption as research has shown that automating is not always superior. The exclusive use of ROI as a calculation of the technology investment value typically results in a naïve estimation. A set of guidelines is provided to enable the test manager to avoid the pitfalls of incorrect application of ROI models. In addition, the study applies the framework to develop the sometimes more meaningful measures of Net Present Value, NPV and Internal Rate of Return, IRR for advanced test technologies as applied for a specific project. The Test Technology Evaluation Framework TTEF is integrated into the NASA IV&V Balanced Score Card. This provides a means of measuring the efficiency of resource allocations for the operational processes of software and systems verification and validation activities that must then be linked to the high level goals of mission success at reduced cost. A measurement framework is necessary to bridge the gap between strategic measures of improved reliability, safety, and quality at reduced cost and operational or tactical measures of optimization of resource allocations applicable to daily activities to achieve these goals.

Future Work

The ISO-9126 Standard documents 6 high-level software qualities including functionality, reliability, usability, efficiency, maintainability and portability. These high-level qualities are mapped to 24 sub-characteristics. Metrics are proposed to measure the high-level software qualities relative to the sub-characteristics. This ISO standard could provide the necessary metrics to measure operational processes under the process aspect of the BSC, relative to the application of product line reuse, and map them to the high-level goals. Of particular interest in this standard is the definition of reusability as the combination of maintainability and portability. It will be of interest to analyze the appropriateness of the standard in measuring reuse for the shuttle.

Specifically, reuse across a vertical product line that incorporates domain engineering, architecture-based reuse, and reusable test technologies.

References

[1] Attewell, Paul, Information Technology and Productivity Paradox, Department of Sociology, Graduate Center of the City University of New York, Report IST 8644358, version 3.1, 1992.

[2] Eickelmann, Nancy S., "A Comparative Analysis of BSC as Applied in Government and Industry Organizations." *Information Technology Balanced Scorecard Symposium*, Antwerpen, Belgium, March 15-16, 1999.

[3] Eickelmann, Nancy S., Evaluating Investments in Emerging Test Technologies. The Proceedings of the *Sixteenth International Conference on Testing Computer Software: Future Trends in Testing*. Bethesda, MD, June 16-18, 1999.

[4] Humphrey, Watts, S., *Managing the Software Process*. Addison-Wesley Publishing Company, SEI Series in Software Engineering, Pittsburgh, PA. 1990.

[5] Kaplan, Robert, and Norton, David, *The Balanced Scorecard: Translating Strategy Into Action*. Harvard Business School Press, Boston, MA. 1996.

[6] Keiso, Donald and Weygandt, Jerry, *Intermediate Accounting*. John Wiley and Sons, USA, 1986.

[7] Strassman, Paul, The Business Value of Computers: An Executive's Guide. The Information Economics Press, New Canaan, Connecticut, 1990.

[8] The Balanced Scorecard Institute, http://www.balancedscorecard.org/default.html, 1999.

[9] Boehm, B., Software Engineering Economics, Englewood Cliffs, Prentice Hall, 1981.

[10] Crosby, P. B., Quality is Free. McGraw Hill, 1979.

[11] Crosby, P. B., Quality without Tears. McGraw Hill, 1985.

[12] Hetzel, B., Making Software Measurement Work. John Wiley and Sons, 1993.

[13] Humphrey, W., Managing the Software Process. Addison-Wesley 1989.

[14] Humphrey, W., Snyder, T., and Willis, R., "Software Process Improvement at Hughes Aircraft," IEEE Software, July 1991.

[15] Jenner, M., Software Quality Management and ISO 9000. John Wiley and Sons, 1995.

[16] Jones, C., Applied Software Measurement. McGraw Hill, 1991.

[17] McGrath, R. and MacMillan, I., "Discovery-Driven Planning" Harvard Business Review, July-August 1995.

[18] Radatz, J. W., "Analysis of IV&V Data" Rome Air Development Center ROME C# F30602-80-C-0115, 1981.

[19] Saiedian, H. and Kuzara, R., "SEI Capability Maturity Model's Impact on Contractors" IEEE Computer, January 1995.

Session 2
SPI and Testing 1

Chairman

Prof. H. Jaakkola

Pori School of Technology, Pori, Finland

THE QUEST FOR QUALITY TEST RESOURCES

• Quality and Efficiency in Software Testing by moving the technological boarder •

Per Jørgensen
Kapital IT, Denmark

About Kapital IT

Kapital IT is situated in a suburb of Copenhagen and has approximately 400 IT-professionals employed.

Kapital IT is a company under Kapital Holding A/S that is the third largest financial institution in Denmark and includes BG Bank A/S and Realkredit Danmark. Kapital IT develops software for these institutions and supports their business domain strategies with IT-solutions.

Kapital IT develops a variety of financial software that is implemented on various mainframe, midrange, and Client/Server platforms. Kapital IT works on a project basis with the emphasis on an efficient development process supported by a development concept that secure consistency in the project. Projects are measured on weather they are carried out as agreed with regard to customer contentment, requirements, quality, economy, stipulated time etc. etc.

The starting scenario

Defect infested software is a serious quality problem. In addition, it is especially a big problem if you still have defects after spending 40% of total development time on test. It is an even bigger problem if your Compass investigation shows that the competitors spend less time on test and at the same time is having fewer defects in production than you.

When your software supports your no. 1 strategic area of growth where the competition is most vigorous and your company continued success on the market rely strongly on the quality of your software - you do have major teeth grinding problems.

The Challenge

We faced a challenge similarly to the once mentioned above. Up through the nineties our Web Banking products grew from small applications into large business and computer complex applications. At the end of the nineties once rather simple applications now involved several platforms of different technical system architecture and furthermore the Web Banking products had increased into a wide variety of products each targeting special customers profile.

At the same time the importance of the products from the business point of view grew and the products eventually became a strategic area of growth for BG Bank A/S and the Danish financial sector as a whole. In the fight for market shares, Web Banking application was now suddenly one of the key factor and of the outmost importance for continued success on the market. It is not an understatement to say that competition was and still is most vigorous.

In order to have a precise picture of our product quality we carried out in co-operation with the international benchmarking firm 'Compass Development' a systematic investigation of productivity, quality, economy etc. in co-operation with systems development and systems administration.

The investigation showed among other things that 30 to 40% of the total time assigned to development was spend on test. This was approximately 5-10% more than companies we compare us self with spend on test.
The investigation also showed that the quality of our products measured in faults per function point was lower than comparable products on the market [1].

To summon up the investigation in one challenging sentence >> we should not use more resources on test, *but use fewer resources more effectively* <<

The Plans and the expected outcome

To *use less resources on test more effectively* is easy to say but hard to do. At least our investigation gave us a clue in what to do. Obviously our development and test process had to be in a degree somewhat 'out of order' and to many defects was implemented into our products, test and production environment. If we could 'fix' our development and test process and find defects early in the requirement phase then maybe we could reduce time spend in the software test phases? To find defects early in the requirement phase we could maybe involve end-users and customers in a different way? Maybe we could automate our GUI tests using modern test tools, maybe that could reduce the defects, and the times spend on test? If we automated the GUI test then why not involve the end-users in that phase too?

Finally, our QUEST began. A QUEST that gave us new knowledge and that lead us to new frontiers. In fact, the QUEST continues to this day and beyond.

The QUEST

Our QUEST began by asking our self the following question >>What if our products suddenly had no (Zero) defects at all and was loved and handled correct by every single customer? <<

If that once became true, several things would happen, for example:
Our products would conquer the entire market
We could close down the 'Hot-Line' situated in BG Bank. No customers would need it.
We could close down our production maintenance crew situated in Kapital IT. No defects equals (almost) no maintenance
We could spare the BG Bank account managers too because the products would need no active sale
Of cause, it was only a very nice dream. Nevertheless, the fact is that better product quality actually does not only create happy customers but also release human resources. Some of which are IT-professionals and others who have in depth business knowledge of various kinds but all with skills that can be used well in the development and test process. You could almost call them The Hidden Resources that just waited to be involved.

We thought a lot about that dream while we investigated and analysed our development and test process [2]. This investigation disclosed the following major opportunities for improvement. We found that:
Our requirement phase was out of focus. We did put great effort into the phase but the real end-user was not present in the phase. Some requirements was not detailed enough or recognised and accepted throughout the company. Still some requirements was not owned and cared for by individual participants.
The end-users was strongly represented in the business domain test but their involvement prior to that was more or less lacking
The customers in the beta test phase would like to participate in the early test phases.
There where plenty of good tools for test automation on the marked. Automated test could reduce test, but we found that the road was paved with automated test that failed. Test automation was possible but required a strong development and test regime before the tools could be used effectively.
To our great pleasure, we discovered the ESSI project, which was willing to support our QUEST experiment.

With these results in mind and still with our dream in fresh recollection we founded our project on 3 major building stones

1. The User and Customer Involvement in Test (UCIT)
2. The Requirement Driven Test (RDT)
3. The Automated GUI Test (AGT)

We baptised the project QUEST for 'Quality and Efficiency in Software Testing' and later we added 'by moving the technological boarder'.

The User and Customer Involvement in Test (UCIT)

The objective of the UCIT phase was to look at The Requirement Driven Test (RDT) and the Automated GUI Test (AGT) to find the optimal way to involve end- users and customers on equal terms with the 'IT-professionals' and thus enhancing the process in general. If possible, it would not be such a bad idea to move the technological boarder between IT professionals and end-users and transfer traditional development activities to the business unit. We could sure use all the IT-professionals we could get. With the national and international shortage of IT-professionals, we just had to use our resources as optimal as possible and that demanded us to rethink the end-users and customers involvement in our development and test process.
From the beginning of the project we choose to look at our end-users and customers NOT as resources in the project but rather to look at what resources that they could bring into the project to strengthen this and the end product.

We invented in our minds a phantom: 'The All European User and Customer'.
Our phantom co-worker 'The All European User and Customer' had no IT education but had lived successfully through the PC revolution. Their IT interest and knowledge was no longer restrained to 'their' application at work. They had a private PC at home. Their IT awareness was high. Some made their own Homepage. Some programmed for fun. They all used the Internet to 'chat', 'surf', use e-mail etc. and it was a long time ago since the phantom 'The All European User and Customer' had second thoughts about IT.
We would love to work together with this phantom and we believed we could find 'it'. And we did!

We found her and him everywhere. In BG Bank they where to be found among product Hot-line staff, among account managers and among banking consultants. Among the customers they where easy to spot. Customers who had been beta-sites several times where almost for sure one of our phantoms. You could almost say that those who would benefit first from better quality where a possible phantom.

They could all contribute to the project with their different in-depth business knowledge. With that and their interest and use of IT we thought that their contribution to the end-product could be more than what end-users and customers ordinarily contributes to in a development and test process. We should soon find out if this was true.

When approached and asked about if they would like to be involve in our project in a different way than usual the vast majority of the 'All European User and Customer' was exited about our ideas. They where excited for different reasons. Some saw the involvement as job enrichment. Some saw new frontiers and job opportunities open up. Others just liked the idea about having influence on the new products design and functionality.

The Requirement Driven Test (RDT)

Our investigation showed us that the requirement phase was out of focus. We would like to get it back on track. We would like to implement into our development concept a RDT phase that made sure that the requirements for new products was detailed to a

level that made test automation possible.
Furthermore, the RDT phase should make sure that all requirements was recognised and accepted throughout Kapital IT and BG Bank.
We would also like if the RDT phase could pin a name on each requirement so each and every requirement had a 'sponsors' with the responsibility to make sure that the requirement was tested and found as described and requested [3].
At last this new RDT phase should of cause involve the 'The All European User and Customer' in reviewing the requirements.

The Automated GUI Test (AGT)

Before our QUEST project, we had some experience with test automation tool [4] [6]. We had even back in the early nineties used these tools in a long period where we converted numerous systems between two platforms. However at that time we found the maintenance task overwhelming and not justifiable from an economical point of view. We had a few GUI test suites left though and occasional we used tools, but only on and on, and off basis. When we used tools, the end-users were never involved because we found the tools to difficult to learn and to use for non IT-staff.
However, we knew that the tools and the tool market had evolved tremendously since we last seriously has 'shopped' for a test tool.
We knew that many companies had automated tests up and running but we also knew that many companies had failed in the automation process.
In spite of this we looked at the market and found that there were many tools to choose from and a lot of them looked surprisingly good *and user-friendly too!*

This gave us an idea. If we could find a tool that not only supported the developer requirements but also was highly user-friendly then why not involve our 'All European User and Customer'? If we together could implement the tool and make our own user-friendly handbook in our AGT process, then why not transfer most of the business domain test from Kapital IT to BG Bank and let the 'All European User and Customer' take control of the AGT process?
If we could move that technological boarder between IT professional and end-users, we could release IT resources for further traditional development. With a better quality, the end-users would be released somewhat from defect handling and this 'spare' time could then be used on the AGT process!

Well it all seemed to fit perfectly. We had the ideas, we had the 3 building stones (UCIT, RDT and AGT) so all we now had to do was to describe the Quest project and then of cause as a minor detail 'sell' the experiment intern in the company and to the ESSI project.
The description part was easy. The 'selling' part took some hard work to accomplish.

The QUEST project objectives

We described the Quest project objectives as follows:

The Quest objective is to improve the test and development process of our multiple platforms that support our Web Banking systems and thus make it more efficient.

The improvements of quality and efficiency is to be carried out through:

Involving the BG Bank end-users in the requirement phase (UCIT, RDT)
Involving the customers in real test and development (UCIT, RDT)
Training of the BG Bank end-users in modern test methods and techniques (UCIT, AGT).
Choosing and implementing a modern test tool that supports business domain test (UCIT, AGT)
Implementing a BG Bank end-user controlled business domain testing process (UCIT, AGT)
Transferring traditional technological test development activities to the business unit (UCIT, AGT)
In re-use the results from the AGT process in subsequent business domain tests of new releases of our Web Banking systems. (UCIT, AGT)

Notice: Our 3 building stones (UCIT, RDT and AGT) mentioned above are tied to the each objective. The Quest objectives were not all that measurable however the objectives were made more measurable in the following Business and Technical objectives.

The QUEST business objectives

The Quest business objectives were as follows:
To reduce the number of production errors by 33% during the first 6 months of a release as a result of a more thorough test and thus strengthen the quality of BG Bank's strategic area of growth
To reduce the production and the maintenance costs by at least 20% do to fewer errors.
To increase the percentage of Hot-Line replies to customer enquiry’s to 98%. This to take place through the staffs strong involvement in the business domain test and because fewer errors means fewer enquiry’s.
Release 10% of the BG Bank end-users in order for them to test future releases of the Web Banking system. This is to take place through fewer errors and thereby fewer customer enquiry’s

The Quest supports the business needs for competitive Web Banking products and gives the customers of BG Bank a quality product at a high technological level. Through this, existing customers are maintained and new customers attracted.

By qualifying the BG Bank end-users to "test super- users" and through their the involvement in the test, we shall obtain a "hands-on" evaluation of new releases before the implementation which will ease the pressure on the business unit in connection with the implementation of a new functionality.

By combining the business domain knowledge of the BG Bank end-users with the system and technical knowledge of the developers, we will achieve a considerable synergetic effect at a critical time in the course of testing. This will have a clear effect on the quality of the final product and will ease the implementation of the new functionality in the area of business.

The Quest Technical objectives:

The Quest technical objectives were as follows:

To form a business domain test of the Web Banking system complex which is end-user-oriented as far as domain-test is concerned and thus releasing at least 10% development test resources for development of new releases of the Web Banking system.
To gain experience and concrete results to be used for subsequent tests of the system complex. This includes test of the readjustment to Economic Monetary Union.
To describe and document a QUEST Best Practice for future and other end-user-oriented tests of the same character in a complex software environment.
To secure that future externally developed operational systems can be implemented in the existing end-user oriented system complex and be tested by means of the "Best Practice" described.
To secure that an automated GUI test process mainly is end-user controlled with quality support from IT professionals.

Notice: The hard numbers or should I say 'the measurable percentages' was founded on the investigation and analysis mentioned in the beginning of the article, the Compass experience and a handful of expectation and 'hope'.

The implementation of Quest

Before we could implement the project, we had some selling to do meaning the project had to be approved by the management in Kapital IT and BG Bank. Finally our quality challenge concerning total time spend on test, faults per. function point etc. mentioned earlier and our thorough investigation and analysis of our current situation did the selling [1] [5]. OK I must admit that it did help that we could say that ESSI and thus the European Commission would finance some part of the project.

Anyway we got the Go Ahead sign and off we went

Implementing 'The User and Customer Involvement in Test (UCIT)'

The UCIT phase was implemented as an integrated part of the RDT and AGT phase. You can actually say that we tried to implement a state of mind where we constantly worked to improve and increase the end-users and customers involvement.
Therefore, it is not correct to look at the phase out of context and not include the other two phases.
Anyway I will give it a try and describe the implementation plan for the UCIT phase in the below 5-step plan:

Find the appropriate activities where the User and Customer can strengthen the project and product the most.
We had already targeted those areas in our project objectives to be a) the requirement phase b) the domain test phase and c) the beta test phase

Find the 'All European User and Customer'
We had already found them in our preliminary investigation. Among those who would benefit the most from better quality and among our beta site customers.

Get them assigned to the project.
When top management approved Quest, we thought that this part would be just a formality. However, we soon discovered that it was not always the case and sometime we had to fight a battle to get the end-users assigned on a serious basis to our project. Of cause when we talk about real customers we had to approach these through official channels and we often found that the customers sales manager where a perfect contact person. He or she had also a clear picture of whether our request would be turned down or not. However, as a golden rule I can say that 'old' beta site customers usually was a sure catch.

Train them well in the activities
Well when you get new staff you train them, right? We did that and kept in mind that our new co-workers had a different approach to IT.

Support them before, during and after
Our new co-workers had ventured into new territories and we found it important to support them in their new role. Before they began to work with us, during the project phases and after when they had returned to their normal work situation.

Implementing 'The Free Test' (UCIT)'

We implemented one activity thou that did not fit under the RDT or AGT 'cap'. It was The Free Test or I could call it the very early beta test. What we did was to release the product for beta test very early in the development phase knowing that it was full of defects. What we actually did was that we more or less finished one part of the product and then released it to our beta sites customers. We told them something like this >>Hey, here is our coming new product! It is far from finish. It is full of defects! Can you find them? These features are almost done; do you like them? What defects do you thing they have? By the way, what you 'put in' you might loose and you can not be sure to upgrade with the next pre release. Please comment on the features and tell us about the defects<<
The readers of this article might thing >>Hey, the morons released a prototype! <<. However, this is not the case. Our pre releases had features that were more or less usable and almost finished. Let us say for example that the customer could do all types of payments but not look at statements. The customer could then get some of the daily work done and at the same time try out the product. Maybe find some defects? Maybe get a good idea for a new or better feature?
We on the other hand had suddenly many customers (read: testers) that did functionality test *and* usability test at same time and for all for FREE.
You might say >>There is no such thing as a free lunch<< and of cause you are right. We had to establish a procedure to support the customers but the cost to run this was nothing compared to the expense if we had had to hire real testers to do the job.
The procedure to run this pre beta test is pictured below.

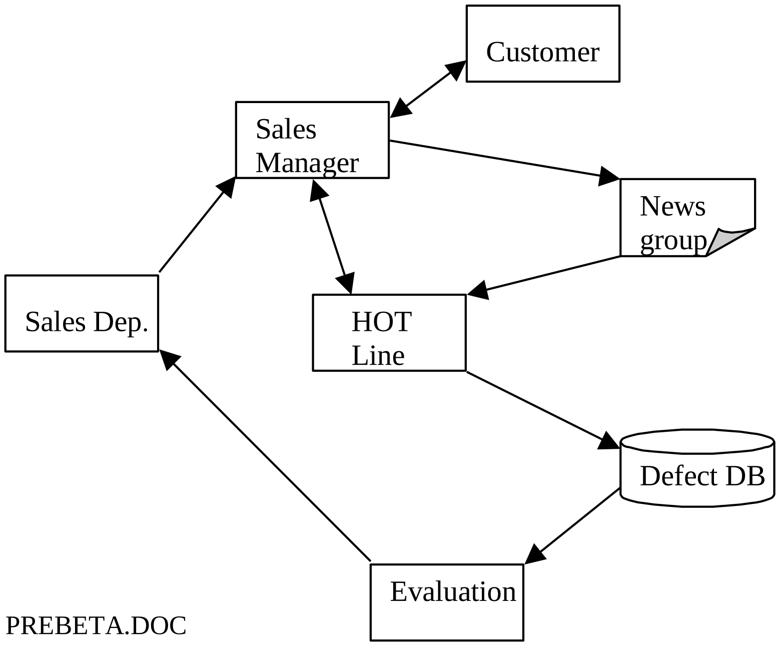


Fig. PEJO.1 : The Pre Beta Test Procedure

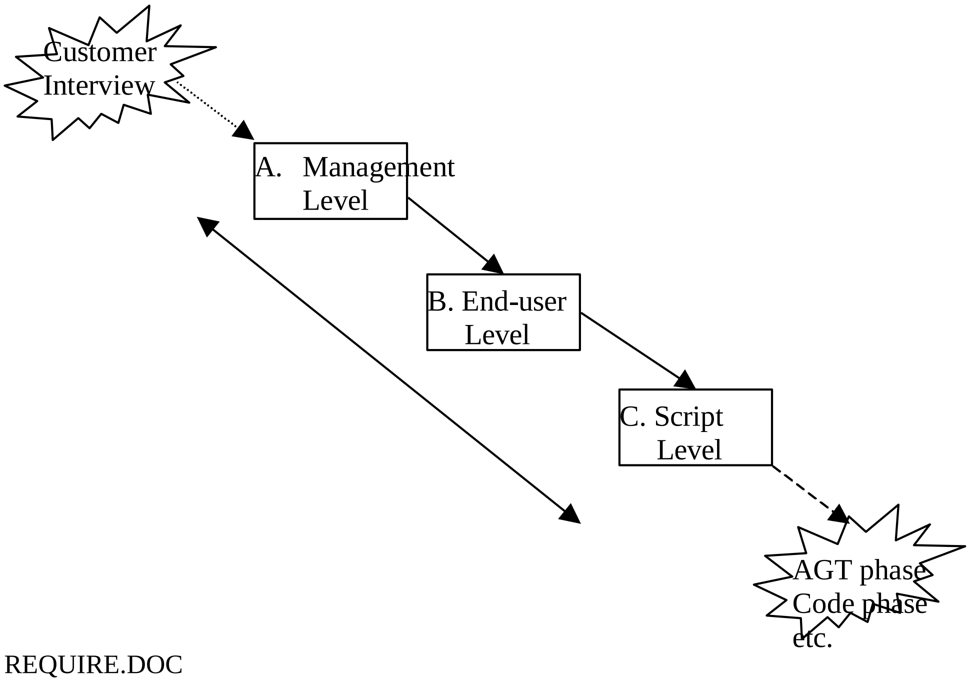
The sales manager is the link to the customer. The sales manager communicates with an established Hot Line or via a news group. Defects and comments are stored in a defect database. At intervals, newly recorded defects and comments are evaluated. The outcome of the evaluation is given to the customer via the sales department and the result is logged in the database.

Implementing 'The Requirement Driven Test' (RDT)'

The Requirement Driven Test (RDT) was implemented as a 3-Level requirement process. Before these RDT levels was established customers was interviewed about their view on existing products, which features they would like in the new product, what they found vital for such a product etc. etc.

This round of interviews spawned or fed some of the requirements for the RDT phase. From the beginning of the phase, we kept a high standard in the documentation of the requirements. We established a database accessible to all participants in the project and kept a strict control to make sure all documentation was linked to the database.

The 3-Level RDT is pictured on the following page:



REQUIRE.DOC

Fig. PEJO.2 : The Requirement Driven Test Phase

We implemented The RDT phase ad a three level process; A) Management level B) End-user level and C) Script level.

At level A. the requirements were identified accepted documented and reviewed. As mentioned earlier some requirements were identified in the interview round with the customer. Other requirements was 'born' as legal requirement, future business requirement etc. at level A. The level contributed at the same time to the framework for the future application cost wise and other wise.

At level B the requirements were analysed described documented and reviewed to a functional level using brainstorms, desk test and usability test. Not only did the end-users and IT professionals participate in this stage also customers were involved through contact with the sales managers. Responsibilities for the individual requirement were anchored in the end-users organisation as well as within Quest project. When the work at level B was finished a formal review was held with participation of management, IT-professionals, and end-users. When the review was completed, the change in the requirements was not possible.

In Level C the end-users broke down the requirements so, it was possible to begin the detailed planning of the automated test. You could say that level C hooked the individual requirement to the automated GUI test. In that way we had control of each requirement, what 'state' it was in etc.

Even thou it was implemented as three separate level the communications between the levels were high and necessary. What went on at one level had to be reviewed commented or otherwise at the other levels.

Prior to the implementation we talked a lot about switching level B to level A and thus having the end-users 'on stage' before the management. Anyway we did not and discovered afterwards that because of the intense communications back and forth between the two levels the discussion prior to the implementation was like talking about what came first: The hen or the egg. Actually we started out with the management but when process got going back and forth, it did not matter much which of the two levels that began the process.

Implementing 'The Automated GUI Test' (AGT)'

Our implementation of the AGT phase were divided into 4 major activities:
Training in test
Selection of test tool
Try-out of test tool and AGT process
The actual AGT process

Training in test
The entire project – IT-professionals as well as end-users - went back to school and were trained in test techniques and test methods. We found it important that all participants had the same basis knowledge of test. Every one went through a tutorial that focused on the test side of the V-model, white- and black box test, preparation of test cases, suites, documentation, end conditions for business domain test etc. etc. The tutorial helped to build some sort of a test foundation in the project from where we could continue down the AGT road.

Selection of test tool
We put a great deal of effort into the selection of the test tool. The tool used for AGT were selected in strong co-operation between the end-users and the developers where the main requirement for the tool was that it should be as user-friendly as possible but also of a high technical development standard [4].
The selection process is described below

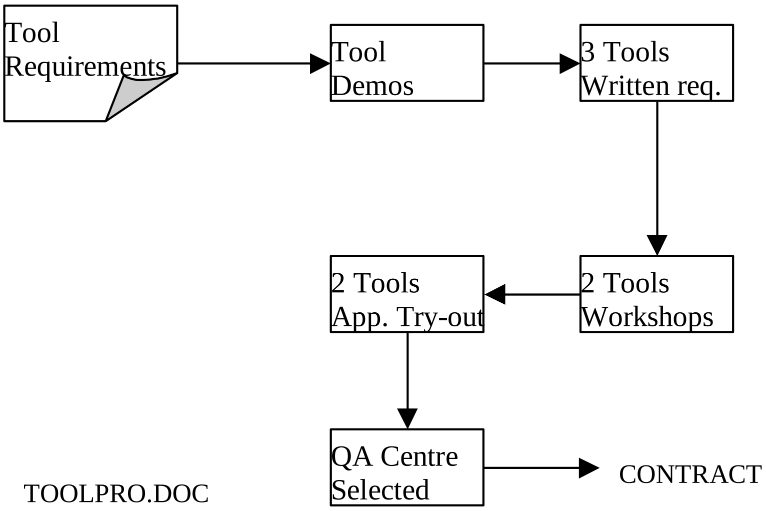


Fig. PEJO.3 : The Tool Selection Process

The project documented the requirements for the tool before a broad selection of tools was reviewed in typical 2 to 4 hours demo sessions. Among these the three most suited tools was selected and the tool vendor was asked to give a written answer to our tool requirement list. The two tools that meet our requirements the best were selected and a

1-day workshop for each tool was arranged. IT-professionals and end-users participated in these workshops and each participant filled out a questionnaire concerning the tool.
Before making the final decision both tool vendors was asked to try-out their tool on one of our web applications. At last we was able to make our choice that eventually was QA Centre from Compuware. This tool was an over all winner and voted the most user-friendly by the end-users and best language by the IT-professionals. Furthermore, we put also a great emphasis on tool support this we also found was best at Compuware.

Try-out of test tool and AGT process
The end-user and IT-professionals was trained in the tools and thereafter was the AGT process including the tools tried out by automating a part of an existing web application. In the try-out, a best practice handbook was established. The handbook describes both the AGT process and how the tool is used in the process. The try-out was a great learning experience and founded the basis to implement Quest in reality as well as in our development concept.

The actually AGT process
A lot has been said and written a lot about automated GUI test [4] [6] [7]. So I will try to outline the differences in the AGT process we implemented. First of all the end-users in our process had been involved in Quest on equal footing with the IT-professionals from the very beginning of the project. They knew the requirements down to every detail and they had prepared the test-cases/script in the RDT phase. They had also extensive tool knowledge and they had tried-out GUI automation in the AGT try-out. Therefore, we implemented an automated GUI test not very different from others like it elsewhere. However, our AGT was to a wide extend end-user controlled. The business domain test was planned and carried out by the BG Bank end-users. Scripts were written *not recorded* by the end-users. Test-suites were build and executed by the end-users. Scripts and the entire test-suite complex were maintained by the end-users. Defects were registered and pre investigated by the end-users. Of cause the IT professionals were also a part of the process but their role was – you could say – reduced to the role as consultant and thus to yield computer, tool and process assistance.

After the implementation of QUEST

After we implemented a great product the end-users returned full-time to their daily work in BG Bank. They took with them the task and responsibility to maintain the test cases, test-scripts and test-suites. At the same time learning-groups were established that meets on intervals where end-users and IT-professionals exchange experience, knowledge, scripts etc. etc.

The Quest Impact, Results and Lesson learned

In our QUEST for Quality we fought numerous battles, some we lost and most we won but most important we won the war for better software quality and we now know that:

The Impact and lessons learned in UCIT phase

The All European User and Customer are great ballplayers in the entire project phase and for sure, it is the undiscovered human resource. The end-user's business domain knowledge strengthens the whole process and ensures a great product. In addition, the Customer involvement makes your product 'stand out'.
Yes! You can move that technological boarder and train non-IT people traditional IT skills and get a better product out of the effort.
When that is stated I can say that we learned that the end-user involvement in the beginning will cost development resources it is not a free ride and you must plan to do a lot of practical work. It is important precisely to describe the objectives and activities of the involvement. Especially when approaching customers it is important to do it in a formal manner. All-ways remember to give responses to customers about what you did with the comments they gave or the defects they found. It is vital to seek and get management consent before all activities and it is maybe necessary to ensure the management consents through the entire project. In involving end-user and customers, it is a good idea to execute each activity in short intensive periods. It is important that end-users return on a regular basis to their base organisation so they can refresh their business domain knowledge. It is important to keep an open dialog about the involvement particularly with end-users not assigned to the project. Treat end-users and IT-professionals a like but be conscious about the end-users non-IT profile.

The Impact and lessons learned in RDT phase

The requirement phase is alpha and omega. You have to manage and test your requirements. You have to have the right people to work with the right requirements at the right time. You have to constantly look critical at your requirement phase and ask your self the question 'Were is there room for improvement? In order to automate you GUI your requirements have to be very detailed – meaning that you have to keep a high degree of documentation. Beware that a high degree of documentation is hard work, it takes and cost time and resources to make and maintain. To document is a boring job to a lot of people. Make sure each requirement is review, documented and that responsibility for each requirement is placed on management level as well as end-user level. Make sure each requirement is pinned (tracked) to you automated GUI test (AGT).

The Impact and lessons learned in AGT phase

Test automation tools are much better than rumoured, some are even easy to learn, and end-users can be transformed into 'tool super test users'. The very difficult part is to build an automated test process that works. You have to implement an ongoing process that exist in every day production supported by well-trained staff and well documented procedures. So yes, you can do regression test, but you will regret and never regress your test suites if you do not have total control of your process.
Initially we planned or expected that the end-user could take control of the entire AGT

phase. Nevertheless, we must admit that we are still is in a 70/30percentage situation where the end-user controls the 70% and the IT-professionals 30% of the business domain test. Not entirely as expected but we game on that most of the 30% can be conquered by the end-users in the years to come. Anyway, we have moved that technological boarder.

The measure of the Quest objectives

By the way: lets have a look at the Quest objectives one more time to see how it turned out in the end:

The QUEST project objectives – Final Score:

The Quest objective is to improve the test and development process of our multiple platforms that support our Web Banking systems and thus make it more efficient.
Involving the BG Bank end-users in the requirement phase (UCIT, RDT)
Involving the customers in real test and development (UCIT, RDT)
Training of the BG Bank end-users in modern test methods and techniques (UCIT, AGT).
Choosing and implementing a modern test tool that supports business domain test (UCIT, AGT)
Implementing a BG Bank end-user controlled business domain testing process (UCIT, AGT)
Transferring traditional technological test development activities to the business unit (UCIT, AGT)
In re-use the results from the AGT process in subsequent business domain tests of new releases of our Web Banking systems. (UCIT, AGT)

Notice: We reached our objectives. Nevertheless, we could have done better in involving the customer. We did a form of usability test in our pre beta test bur I am afraid to say that we started out with greater plans for usability test. However the requirement phase ate more of our time and resources that planned and therefore we had to scale down the extent of the usability test.

The QUEST business objectives – Final Score:

To reduce the number of production errors by 33% during the first 6 months of a release as a result of a more thorough test and thus strengthen the quality of BG Bank's strategic area of growth
To reduce the production and the maintenance costs by at least 20% do to fewer errors.
To increase the percentage of Hot-Line replies to customer enquiry’s to 98%. This to take place through the staffs strong involvement in the business domain test and because fewer errors means fewer enquiry’s.
Release 10% of the BG Bank end-users in order for them to test future releases of the Web Banking system. This is to take place through fewer errors and thereby fewer customer enquiry’s

The QUEST Technical objectives – Final Score:

To form a business domain test of the Web Banking system complex which is end-user-oriented as far as domain-test is concerned and thus releasing at least 10% development test resources for development of new releases of the Web Banking system.
To gain experience and concrete results to be used for subsequent tests of the system complex. This includes test of the readjustment to Economic Monetary Union.
To describe and document a QUEST Best Practice for future and other end-user-oriented tests of the same character in a complex software environment.
To secure that future externally developed operational systems can be implemented in the existing end-user oriented system complex and be tested by means of the "Best Practice" described.
To secure that an automated GUI test process mainly is end-user controlled with quality support from IT professionals.

Notice: To be total honest I must admit that we are still in our measuring phase. So far it seems that we will reach the objectives and for some objectives beyond them.

The QUEST continues

So we can proudly say that we have a test process that involves and rely on The All European User and Customer - and it works! We did not use more resources on test than expected or as used on previously projects. We did implement a great quality product almost defect free. We did implement the product on time and our cost on maintaining the product equals our competitors.
Our QUEST has begun and it will continue always. We can and we will do it better, faster and cheaper. We still carry out four annual investigations of our system development process to constantly keep the focus on activities to improve. For the moment we work on a true ‘Release when you please’ process. This combine the strength of our automated Client/Server test with the ‘On the fly’ customer requests and demands that ties the Client/Server, midrange and mainframe together in a test process that enables new releases to be build ‘over night’.

References

[1] Jones Capers, Applied Software Measurement: Assuring Productivity and Quality. On the history and evolution of functional metrics pp. 43-122, McGraw-Hill, Inc. (New York, N.Y.) 1991.

[2] Ould Martin A., Strategies For Software Engineering: The Management of Risk and Quality. On the 14 dilemmas of software engineering, pp 186-223, John Wiley & Sons, Inc., (New York, N.Y.) 1987.

[3] Hertzel Bill, The Complete Guide to Software Testing. On testing methods and tools, pp 43-72, John Wiley & Sons, Inc., (New York, N.Y.) 1988.

[4] Vinje Poul Staal, Test af Software. Planlaegning og styring, pp. 253-320 and Vaetoejer pp. 359-381, Teknisk Forlag, 1993.

[5] Zahran Sami, Software Process Improvement. On launching implementing and measuring software process improvement, pp 183-232, Addison Wesley Longmann, 1998.

[6] Fewster Mark, Grove Consultant, The Selection, and Use of Test Execution Automation Tool, 2 day Seminar at Delta, 1997

[7] Kolish Tony, Segue Software Inc., Ensuring Reliability of Web Applications: A white paper for IT managers, QA professionals and software engineers. The paper was presented at the EuroStar 1997 conference in Edinburg.

Software Inspection Techniques in SMEs

Francisco J. Rodríguez
ELIOP,Madrid, SPAIN

Manuel Villalba
ELIOP,Madrid, SPAIN

Ignacio González
ELIOP,Madrid, SPAIN

Introduction

The TESIS project is aimed to use software inspection techniques in the context of an SME.

Software inspection is generally appreciated as a method to improve the software product quality and, by decreasing the amount of rework, to reduce development time and costs and to increase productivity. However, the main purpose of an inspection process should be to supplement testing, not to replace it, having in mind that testing alone will not determine if code will work on different platforms, if it is written efficiently and whether it adheres to particular coding guidelines or standards.

Reported inspection results vary considerably, although all the reports claim that the use of this method, and its variations, improves product quality. Some experiences have established a capability of software inspections to identify up to 80 percent of all software defects early during the software development stage [7, 11, 13]. Other teams in AT&T mention that the percentage of defects removed through code inspection varies widely from about 30 to 75. After improving the inspection process, they achieved defect-removal efficiencies of more than 70 percent [2]. Moreover, when inspections are combined with normal testing practices, defects in fielded software can be reduced by a factor of 10. These reasons, together with the productivity increase and the reduction in costs and delivery time, explain the use of these methods and techniques by a variety of manufacturers.

Other forms of inspection, based on the Fagan’s one, have been developed. The Jet Propulsion Laboratory (JPL), California Institute of Technology, tailored Fagan's original process of software inspections to conform to its software development environment in 1987. Also, AT&T Bell Laboratories refined the original inspection process through the setting of a measurement system that defines nine metrics to help plan, monitor, control and enhance the inspection process. Innovative techniques, like Collaborative Inspection methods and tools, have been introduced with the aim of efficiently supporting the inspection process [6], and an interesting variation of inspection, denominated N-fold inspection, uses traditional inspections of the user requirements document but replicates the inspection activities using N independent teams [10]; this parallel technique has been used in the development of mission-critical software systems.

It seems mandatory to say that inspections are focused on finding defects, neither to correct them nor to improve the software product. A variation of the Fagan's inspection process adds an extra step after the meeting for discussing corrections and general improvements [8]. Also, team size varies among three to five members, being possible teams formed by only two developers [3].

The SPICE ISO 15504 standard (v. 2.0) covers software inspections in several sections. Although SPICE doesn't fix specific procedures to carry out inspection activities, there are processes related to inspections in Support, Management and Engineering categories.

Regarding CMM (Capability Maturity Model), a key process area more directly related is undoubtedly *Peer Reviews* (PR), established in level 3 of the model, being the only software engineering specific procedure showing such importance. The reason of this exception is the general agreement about the effectiveness of this method in the early detection of defects, although this CMM concept is not so formal as Fagan’s inspections.

Regarding the several stages in a software inspection process, planning, overview, preparation, examination, re-work and follow-up, the importance of each of them varies according the authors. Overview is usually considered as optional, specially if the inspection team is composed by skilled developers familiar with the project and the software product. However, the point of view of Ackerman et al. [1] is that overview and preparation are often underused.

Christenson [5] relates an experience in which defect data were collected over several years enabling estimation of the number of defects in uninspected code, inspection effectiveness, and the number of defects remaining after inspections. In addition, the authors were able to derive an optimal level of inspection effort, given a work product initial defect density. According to that estimation, some decision could be made about re-inspecting the code after the rework phase or continuing on into the test phase. The cost of finding defects by inspection was sufficiently low that even products with relatively few defects on first inspection were reinspected. Products with many defects on first inspection were already candidates for reinspection.

Regarding initial training, a comprehensive guide about software inspection [9] has been taken during this project as a fundamental reference to initiate the navigation through the software inspection processes world.

Although most of the effort has been obviously put into tuning the existing techniques according to ELIOP size, projects and culture, the TESIS project aims to extract interesting results for the wider community, because there is a relatively small amount of reported experience related to Software Inspection in SMEs.

Applicability of software inspection techniques at ELIOP

The company

ELIOP is a manufacturer of hardware and software products in the market of supervision and control systems for industrial applications. People involved in Software Engineering within ELIOP are grouped in several areas, and, within each area, a team is created for each project. The current engineering practice at ELIOP is oriented to the classic life cycle steps: requirement analysis, design, coding, testing and maintenance.

The projects developed within the company range a wide variety of different types and sizes: software with hard real-time constraints for embedded systems, software for man-machine interfaces (MMIs), SCADA (Supervisory and Data Acquisition) systems to supervise and control distributed networks, etc. Most of these systems must work continuously at remote unattended locations, where modifying or updating the software is costly and difficult.

Among the new projects, many of them can be considered upgrades of already existing products. These projects consists on adding new software to these products or modifying them to cover a wider range of cases. A mixture of high skilled software engineers and less experienced people composes the baseline projects teams.

Starting scenario

In the last years, ELIOP has assessed its practices because of several facts:

ISO-9001 certification for all its processes, including software design. Procedures according to ISO-9001 are being applied to the software activities from the beginning of 1995.

Participation in two process improvement experiments supported by the ESSI (10396 ISORUS and 21222 AMIGO).

These circumstances, as well as the metrics data obtained as a consequence of the above mentioned PIEs, have strongly influenced the practice of software production at ELIOP, being as follows at the starting time of the project:

Documents associated to each phase must be formally approved by the project management before starting the next stage. As a consequence of previous ESSI projects, a "Domain Analysis" step has been established previous to the requirement analysis. Also, software reuse practices are systematically considered in the remaining steps.

Special attention is paid to the requirements specifications. They are normally written in natural language, with some graphical support. No formal languages are used, nor tool-assisted tracing of requirements through the life cycle. However, according to the results of the ESSI project AMIGO, inaccurate or incomplete specifications can cause up to 20% of maintenance problems.

Structured analysis and design and object-oriented design techniques are used in some projects to support the design phase. In most projects, design is documented without formal notations, by means of code decompositions, and processes, data and interface descriptions. Two different languages, C and C++, are normally used for programming, although legacy code exists written in a variety of other languages.

Configuration management policies are applied to source code and some other software assets in a systematic, tool supported way.

No software inspection practice was running before the beginning of this project.

Metrics data are collected, most of them related to the efforts devoted to each project, and to the number of software problems reported. These data show that the projects with higher deviations from schedule have frequently only small deviations up to the end of the coding phase, and most of their deviation comes from repeated testing-rework cycles performed after code is finished. When a project is affected by such a problem, hidden lack of quality appears very late in the life cycle, when efficient corrective actions are difficult to undertake.

The analysis of the causes of the software problems appearing after delivery to the customers reveals that inadequate specifications, design, coding and testing contribute with similar weights to the appearance of defects. Accordingly, all these phases should be subjected to inspection in order to decrease the number of defects transmitted from one life cycle phase to the next one.

The baseline projects of the experiment

Two well different projects have served as baseline projects to carry out the TESIS experiment:

A medium sized project (about 100 KLOC of code plus 150 KLOC of testing code) consisting on a new Embedded Control System with demanding requirements in reliability and safety. This project involves both hardware and software development and is performed by a team of six software engineers. The software is mainly written in C, pursuing real-time performance. This project has just been started at the beginning of the PIE and is planned to finish within 1999.

The other baseline project consists of a small project whose objective is to add a new relevant function to an existing Remote Terminal Unit (RTU). ELIOP usually needs to undertake these type of projects, modifying existing systems, as part of its normal operation. The software product embedded is a real-time one, written in C and mainly composed of a reused and modified code. The project team is normally composed by two or three engineers.

There exist several reports about inspection experiences over safety critical systems. Martin's paper [10] is mainly centred on a kind of parallel inspection, accomplished by some independent teams, of the user requirements documents. Tripp’s paper [12] discusses also the application of multiple team inspections to improve the technical review and quality of a safety-critical software standard.

Curiously, it seems more usual to inspect documents than code when talking about safety critical software, although Buck’s [4] report discusses the result of the application of the inspection process to software developed at IBM for the United States Navy. In data analysis they have found certain very consistent values in their environment. For example, their new code has about 8 to 12 defects per KLOC and they require about 3 to 5 man-hours per major defect detected in inspections. A major defect is defined in terms of interfering with program performance.

Regarding real-time characteristic, the AT&T experience mentioned above [2] is primarily devoted to real-time embedded systems written in C by teams of 3 to 80 developers. The projects inspected new, modified, and reused code.

The amount of information encountered about similar projects has been considered enough to take references during the initial phase of the TESIS project.

The skills of the staff are not exactly the same. Some of them are highly skilled personnel, while others, although not novices, have a shorter professional experience. The role of moderator is expected to be assumed by the most experienced engineers belonging to the team. Changing the role among the team members is foreseen as a method to dynamically extend the software inspection culture. Anyway, all of the co-workers are novices on Software Inspection techniques.

Due to the fact that one of the baseline projects is a small one, its inspection team size will be composed, as maximum, of three members. One of them will assume the roles of moderator and recorder at the same time. The possibility of accomplishing two members inspection is being studied. An interesting fact is that, according Bisant paper [3], this method appears more effective at improving the performance of the slower or novice programmers.

On the other hand, the inspection teams corresponding to the big project will be composed of three or four members, in order to extract conclusions about inspection efficiency and associated costs.

Expected outcomes

The expected results of the TESIS project are the following:

Reduction of the defects reported by the customers. The established target is 30% reduction in two years. Identification and removal of systematic defects.

Increased development productivity, due to lower development costs and less time dedicated to remove defects reported in delivered software.

Better achievement of project schedules by achieving repeatable and normalised software production processes. The goal is 50% reduction of the current average deviation in two years.

Rapid cross-training of software engineers participating in a project.

Indirect benefits on team building.

Additionally, the reduction of the number of defects existing in delivered products necessarily implies the increasement of the satisfaction of the customers and the enhancement of the image of the company.

The quantitative objectives can only be measured when the software inspection processes will be widely used in the company. The accomplishment of objectives mentioned above for the baseline projects will be shown within the PIE based on the measure of inspection efficiency. According to our preliminary estimations, an efficiency better than 66% should allow to achieve the defined objectives.

Implementation of the improvement actions

To accomplish the improvement actions, apart from the project management, training and dissemination actions, the TESIS project has been divided into several tasks or workpackages.

A General Approach Study established the basic approaches to carry out the experiment, analysing the applicability of software inspection techniques to the selected baseline projects. Also, adequate directions for subsequent project activities were identified after studying the available documentation.

After completing the first stage, the existing scenario at ELIOP was revised as well as the required process changes. This was the origin of the Internal Procedures document, written in order to support all the software inspection activities to be accomplished within the context of the company software development process. These Procedures include a description of the process, a detailed description of the inspection activities, a definition of the metrics to be applied and all the necessary checklists and guidelines to be used along the inspection process. Also, inspection forms are included, for making easier the collection of all metrics data to be stored in the inspection database.

Metrics identification and set-up was the next step of the project. The different metrics included in the technical Procedures were identified and selected during this workpackage. The intended goals pursued by the metrics were established and translated into measurable magnitudes, and a metrics plan was designed to collect all the needed data.

Following the Internal Procedures set-up and metrics definition, the experimentation activities started for the baseline projects, even earlier than the foreseen date due to the project needs and scheduling. The first baseline project, devoted to the production of safety critical software, includes software inspection as a regular practice to assure the software quality. At the time of writing this paper, the baseline project is already running and the results obtained have been quite good, as well as the acceptance of the inspection techniques among the engineers participating in the project. Regarding the second baseline project, source code inspections have been done although, being this project a small one, the results are not so significant than those corresponding to the first project.

For supporting the experimentation activities, a number of tools were evaluated and thoroughly tested. Finally, no specific tools were purchased, being preferred general tools for helping in the navigation and annotation of source code.

Internal procedures are dynamically revised, using the data taken and the experience gained from the experimentation. An external subcontractor and ELIOP management participates in this review, with the objective of obtaining a revised set of procedures ready for its introduction in the regular practice.

At the time of writing this paper, the evaluation of the experiment has not completely finished, although most of these task results can be considered as final ones. Specifically, this workpackage deals with the comparison of the global results of the project with its original objectives, the analysis of all aspects of the project and the identification of the lessons learnt.

Finally, an introductory plan will be written, to introduce the identified changes in the normal software development process at ELIOP.

Results and lessons learnt

Metrics applied

Previously to the experimentation phase, an Internal Procedures document for carrying out inspection activities was approved. Even before writing the document, the inspections team gained some experience through trials and a training activity.

To choose the adequate metrics is essential for the success of the inspection process. In this sense, we have followed the Barnard approach [2] applied in ATT and derived from GQM. The first step was to clearly establish the intended goals pursued by the metrics plan. After being established, these concepts have to be translated into measurable magnitudes and a metrics plan designed to collect all the data needed.

The identified measurement goals were the following:

Plan software inspections as part of the whole software development process.

Monitor and control the inspection process.

Monitor and control the quality of the inspected software product.

Monitor and control previous stages of the software development cycle.

Improve the inspection process.

As we can notice, only the third and fourth points are dedicated to directly highlight the benefits of software inspection activities. The other points are focused more on the control and continuous improvement of the inspection process. Due to that reason, care has been taken in the adoption of these metrics to evidence the benefits provided by software inspection versus the associated costs.

Defects have been classified, at least, as major or minor defects. Major defects are those which affect the functionality of the software product under inspection. Minor defects are those which, not affecting the product functionality, affect in a way or another to the product quality or indicates a lack of conformance with the software development rules established by the company.

It is worth to clarify that defining how to exploit metrics data is as important as defining the data. A simple step by step procedure was established for this purpose:

Inspection data have been stored and collected in an inspection database. Microsoft Access has been selected for storing software inspection data and for elaborating inspection reports, due to its wide availability. The relational database used has been a very simple one, in order to avoid wasting of time in its implementation.

Inspections trends were analysed over time, and control applied to maintain metrics results consistent with project guidelines. Each analysis should compare the current results with valid references or an average of past results, depending on the specific metric.

Inspection or development processes have been adjusted according to metrics results when indicating a systematic problem.

Finally, plots of metrics data have been used to graphically highlight hidden trends.

During the experimentation phase, the following metrics have been applied to the data collected from inspections:

Metrics	***Formula***
Average effort per inspection unit (time in hours / LOCs, pages, test cases)	$\frac{\sum_{i=1}^{N}(\text{Inspection effort i})}{\text{Inspected units}}$ N = Number of inspections Inspection effort *i* = Planning time + Preparation time + (Inspection duration x Number of participants) + Rework time + Revision time.
Percentage of reinspections.	$(\frac{\text{Number of reinspections}}{\text{Total number of inspections}} \times 100)$
Average inspection rate (LOCs, pages or cases / time)	$\frac{\text{Inspected units}}{\sum_{i=1}^{N}(\text{Duration of inspection i})}$ N = Number of inspections
Cost per removed defect.	$\frac{\sum_{i=1}^{N}(\text{Inspection effort i})}{\sum_{i=1}^{N}(\text{Number of defects detected during inspection i})}$ N = Number of inspections Inspection effort *i* = Planning time + Preparation time + (Inspection duration x Number of participants) + Rework time + Revision time.
Average preparation rate (LOCs, pages or cases / time)	$\frac{\text{Prepared units}}{(\sum_{i=1}^{N}(\frac{\text{Preparation time i}}{\text{Number of participants in preparation i}}))}$ N = Number of inspections
Number of detected defects per inspection unit.	$\frac{\sum_{i=1}^{N}(\text{Number of defects detected during inspection i})}{\text{Inspected units}}$ N = Number of inspections

Other metrics are foreseen to be applied but, at the time of writing this paper, there have not been collected enough data to permit their direct application.

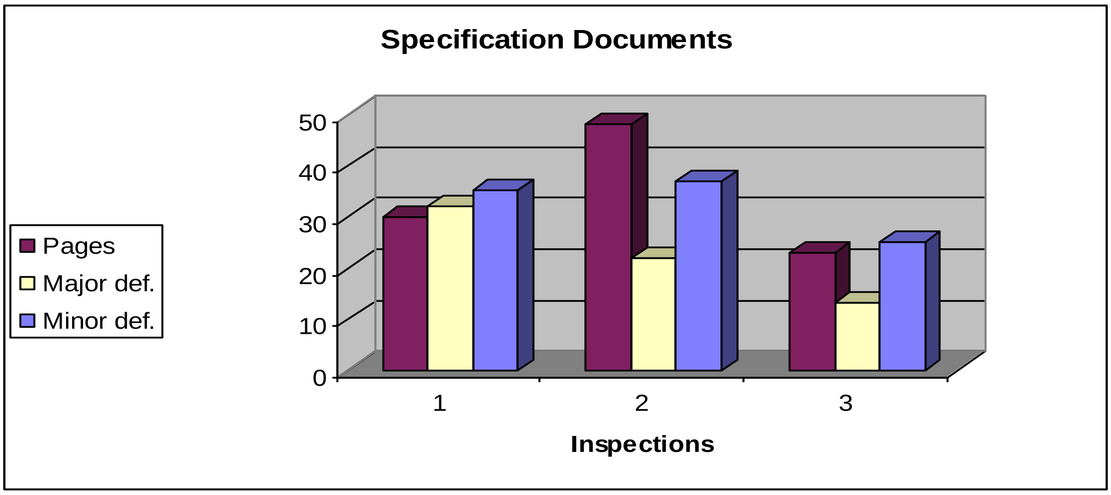
Software requirements and design documents results

As commented above, both specification of requirements and design documents have been inspected, although the majority of the inspections have been focused in encountering defects in source code.

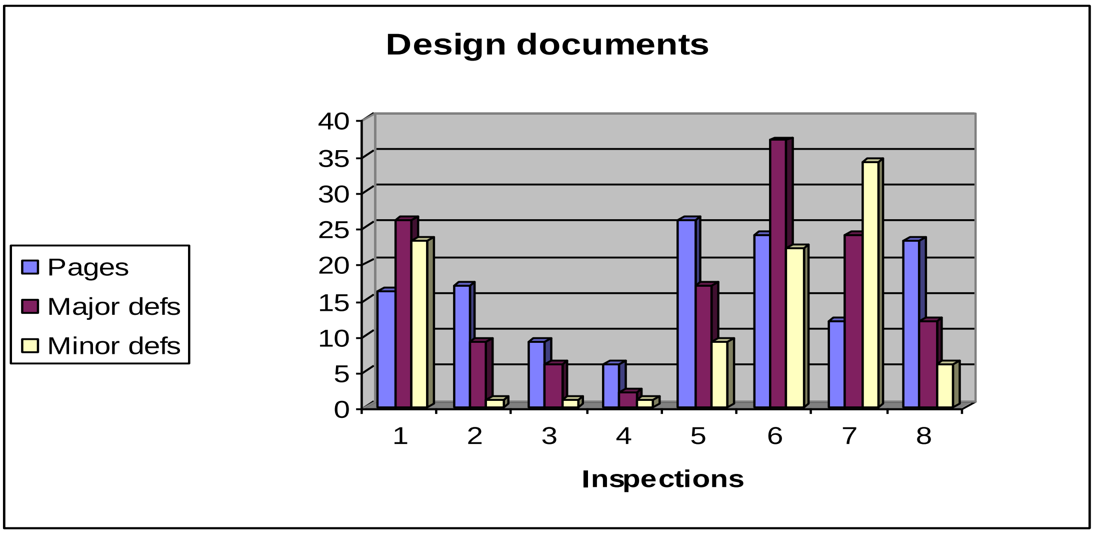
Three documents of specification of requirements have been inspected. Among them, the Functional Specification of Requirements of the biggest baseline project, regarding safety critical software.

Within the documents for specification of software requirements, no reinspections have been produced. Due to the importance of the inspected documents, it seems that, perhaps, it is worth to inspect again the documents if an enough number of defects has been encountered during the first inspection. But, in the case of the Functional Specification of Requirements document, the only document with an important number of detected errors, a major revision of the document occurred after the first inspection and due to the external reasons. So, a completely new version and the document was generated and inspected. We have considered this process more a first inspection than a reinspection.

The average number of defects per page is 1.71, although the relation among major and minor defects varies from one inspection to the others. The figure below shows the number of major and minor defects and the number of pages corresponding to each inspection.



Regarding inspections of design descriptions, seven documents have been revised, one of them along two inspection meetings, due to its size.



Similar conclusions can be extracted for design documents than for specification of requirements. The average number of defects per page is 1.75 (approximately 1 defect per page for major defects and 0.75 for minor defects).

Both for specifications and design documents, typewriting errors have been removed from the documents prior to be subjected to the inspection process.

Code inspection results

Up to the moment of writing this report, 49 inspections of code have been carried out. Among them, only 4 have been re-inspections: that is to say, the percentage of re-inspections is approximately 8%.

Regarding code inspections the average effort per inspection unit is 0,014 per LOC or, in an equivalent way 14 hours are devoted to complete the whole inspection process of one KLOC (including the planning, overview, preparation, rework and review phases). See below a table detailing the effort spent along the different inspection phases per inspection unit.

	Average effort (hours)	**Average effort per inspection unit (hours/KLOC)**
Planning	1.5	0.3
Preparation	15	3
Meeting	29.5	6
Rework	10.5	2.5
Review	3	0.6

while 409 LOCs are inspected each hour during the inspection meeting by the development team. These figures are of relevance for planning the effort to be dedicated to code inspection along the lifecycle of a specific project. However, it has been observed that the last figure, Average inspection rate, can widely vary, mainly due to the inspectors experience, not only in coding, but also in the inspection process itself.

Sometimes, when the inspection team is composed of only two members and one of them adopts all the roles with the exception of “author”, the number of inspected units per hour of inspection meeting significantly increases; that’s to say, an inspection team of two members seems to inspect considerably faster than a bigger team. Moreover, this effect doesn’t imply a reduction of the inspection efficiency. However, this team composition is only possible when the “super-inspector” is a very experienced developer and very familiar with the development of the revised modules.

The cost for removing a defect is 0,28 hours, although this figure is due more to minor or non-functional defects than to major ones. If we consider only major defects, the cost increases up to 1,48 hours.

The average preparation rate is 265 LOCs per hour, being this figure approximately constant.

Finally, the number of defects encountered per KLOC is 36, including both major and minor defects. If we only want to consider major or functional defects, the value decreases up to 6.

Test cases results

With regards of test case inspections, the total number of inspections ranges 7, and no module has been reinspected.

In average, 0,006 hours are devoted to revise one inspection unit (in this case, the inspection unit is an element of the test case tables), and the average inspection rate is 841 units per meeting hour.

The cost figures are similar to the ones corresponding to code inspections and the above commented trends also remain. The costs for eliminating a defect is 0,87 hours, being this figure influenced more by minor defects than by major ones. Considering only major defects, the costs increases up to 3,65 hours.

The average preparation rate is 824 test case elements per hour, and the number of defects encountered per element is 0,006, including both major and minor defects. For major defects only, this figure decreases up to 0,0015.

Lessons learnt

Regarding a technical point of view, it is worth to remark the following lessons learnt during the project life:

Software inspection is a very effective way of finding defects in an early phase of the product life, but it is also a costly method. Due to the last reason, care must be taken in applying these techniques in a controlled way and demonstrating the inspection benefits to the company management.

Source code inspections usually serve also to detect, in an early phase, design defects, so it is essential to start source code inspections as soon as possible.

During the first inspections, the adoption of the role of the moderator by the technical team head facilitates, due to his high degree of expertise, the cross-training of the inspectors, even more regarding key aspects of complex projects.

Inspections are much more efficient when the preparation phase and inspection meetings are held immediately after finishing the implementation of the software product.

The "third hour" meeting is really important to achieve a better knowledge of the entire project for the software engineers participating in it. Also, it is an excellent training tool for the novices in the project.

Moreover, due to its intended formalism, it is important to clearly define the minor aspects of software inspection technique in order to avoid any waste of time, and to decrease the effort dedicated to inspections. The internal procedure document must be comprehensive and quite detailed.

Also, organisational aspects, like cross-training and team building, constitute an indirect but clear benefit of software inspections.

References

[1] Ackerman, A.F., Buchwald L. and Lewski F.H., Software Inspections: An Effective Verification Process in: *IEEE Software*, Vol. 6, No. 3, pp. 31-36, 1989.

[2] Barnard, Jack, Art Price, Managing Code Inspection Information in: *IEEE Software*, Vol. 11, No. 2, pp. 59-69, 1994.

[3] Bisant, David B. and James R. Lyle, A Two-Person Inspection Method to Improve Programming Productivity in: *IEEE Trans. Software Eng*., Vol. 15, No. 10, pp. 1,294-1,304, 1989.

[4] Buck, Robert D. and James H. Dobbins, Application of Software Inspection Methodology in Design and Code in: *Software Validation*, H.L. Hausen, ed.,Elsevier , pp. 41-56, Amsterdam 1984.

[5] Christenson, Dennis A. and Steel T. Huang, A Code Inspection Model for Software Quality Management and Prediction in: *Proc. GLOBECOM '88. IEEE Global Telecomm. Conf. and Exhibition*, pp. 468-472, 1988.

[6] Drake, Janet et al., Support for Collaborative Software Inspection in a Distributed Environment: Design, Implementation, and Pilot Study in*: Tech. Report*, TR 92-33, Univ. of Minnesota, June 1992.

[7] Fagan, Michael E., Design and Code Inspections to Reduce Errors in Program Development in: *IBM Systems J.*, Vol. 15, No. 3, pp. 182-211, 1976.

[8] Gilb, Tom, Principles of Software Engineering Management, *Addison-Wesley*, Mass., pp. 205-226, and pp. 403-422, 1988.

[9] Gilb, Tom and Graham, Dorothy, Software Inspection, *Addison - Wesley*, 1993.

[10] Martin, Johnny and Wei-Tek Tsai, N-fold Inspection: A Requirements Analysis Technique in: *Comm. ACM*, Vol. 33, No. 2, pp. 225-232, 1990.

[11] Myers, Ware, Shuttle Code Achieves Very Low Error Rate in: *IEEE Software*, Vol. 5, No. 5, pp. 93-95, 1988.

[12] Tripp, Leonard L., William F. Struck, and Bryan K. Pflug, The Application of Multiple Team Inspections on a Safety-Critical Software Standard in: *Proc. 4th Software Eng. Standards Application Workshop, IEEE CS Press*, pp. 106-111, Los Alamitos, Calif, 1991.

[13] Weller, Edward F., Lessons from Three Years of Inspection Data in: *IEEE Software*, Vol. 10, No. 5, pp. 38-45, 1993.

Company profile

Founded in 1979, ELIOP is a Spanish medium size industrial enterprise with a subsidiary company in Turkey. With 140 employees, 80 of them having a University degree, and a turnover of 12 million ECUs, ELIOP is very active in the domain of Information Technologies.

ELIOP is a Hardware and Software Factory. Its products, entirely developed within the company, include Remote Terminal Units, for Telemeasurement and Telecontrol applications, large distributed Supervisory Control and Data Acquisition systems (SCADA) and also Computer Vision systems. These products are integrated in turn-key systems that the company sells in national and international Electricity, Transport, Gas, Petrol, Water, and Environment markets.

ELIOP is offering innovative solutions for these markets not only in Spain, but also in Europe and Mediterranean countries. Competing with the most important international companies in the transportation and energy sectors (mainly from the USA), ELIOP is developing relevant projects in many Latin America countries: Brazil, Colombia, Ecuador, Peru, Argentina, Paraguay, etc.

A significant part of the ELIOP yearly budget is devoted to Research and Development activities. The company participates in several national and international RTD projects. It has taken advantage for improving its software processes through the following ESSI projects: ESSI Project 10936 ISORUS, ESSI Project 21222 AMIGO and ESSI Project 27506 TESIS.

Curricula

Francisco J. Rodríguez

Born in Madrid (Spain), 28th June 1963. Physicist, specialised in Computers and Automatic Control, Complutense University of Madrid.
From 1988 to 1991, he worked in the CSIC (Consejo Superior de Investigaciones Científicas) in the field of ultrasonics, signal processing and automatic data acquisition systems. After five years working in the field of automatic systems for NDT (Non Destructive Testing) (TECAL, 1992 to 1997), he joined ELIOP, first as ELIOP project manager of ESPRIT projects 8819 VICTORIA and ESPRIT INNOVA 21017 and, from March 1999, as responsible for the Software Area within the R&D Department.
He is the author of several international publications in the field of ultrasonics and automatic systems for NDT.

Manuel Villalba Quesada

Telecommunications Engineer (Universidad Politécnica de Madrid) 1978. During his professional career, he has worked in Telefonía y Electrónica, SA. (1979-1980) in the hardware design of telephone subscriber equipment, and TELETTRA España, SA. (1980-1983), R&D division, as team head for hardware and software design of special telephone equipment.

Since 1983, he is working at ELIOP, SA., participating as technical manager in many projects in the industrial electronic equipment and systems field.

Also, he has participated in several projects funded by the European Commission, among them the ESPRIT projects 5184 LOCOMOTIVE and 8819 VICTORIA, and the ESSI projects 10936 ISORUS and 21222 AMIGO as Project Manager. He is currently in charge of the management of the ESSI project TESIS.

He has been until recently Manager of the Software Area in the R&D department. He is now Marketing Manager for ELIOP product lines.

Ignacio González Torquemada

Born in Madrid (Spain), 4th August 1963. Industrial Engineer, specialised in Electricity, Electronics and Control. E. T. S. Ingenieros Industriales de Madrid, Universidad Politécnica de Madrid, 1987.

He has worked in DISAM (U.P.M. University, 1987), programming vision-based industrial systems. From 1987 to 1989, he worked in ELIOP S.A. (1987-1988) developing real-time PC programs, and TID S.A. (1988-1989) as an applications engineer and hardware designer.

Since 1989, Ignacio González is working at ELIOP S.A., first as responsible for SW Development of Automation, Tariff and Electric Load Control products and, after that, as SW Group Manager for Automation and Interlockings.

He has worked in several ESSI projects: ISORUS (Software Reuse), AMIGO (Software Maintenance) and TESIS (Software Inspection Techniques).

He is the co-author of the paper “A sequential edge detector using edge and grey level histogram information (1988), International Workshop on Sensorial Integration for Industrial Robots (SIFIR’89). Zaragoza, Spain.

Productivity Improvement via Software Testing

Martin Prieler

EDV Ges.m.b.H, Vienna, Austria

Introduction

Back in 1997 EDVg was participating in a European System and Software Initiative (ESSI) task called PIE (Process Improvement Experiment). This task is mainly dealing with best practice in the field of software engineering and is founded by the European Commission (EC).

Our proposal was accepted and we were starting our PIE on June 1st last year with an overall project duration from 15 month our estimated project end is 1st of September 1999. One of our obligation within this experiment is to disseminate our results on at least two international conferences dealing with a similar background. We already made our first presentation on the 6th European Conference on Software Quality [1] in Vienna earlier this year. Now we are going to present our final results and findings on our 2nd and last international presentation.

The goal of the PIE was the introduction of an already proven testing concept, its evaluation in practice (putting it into action on a real software development project), its support by suitable tools, and the adaptation (optimisation) of this concept based on the lessons learnt. This concept is part of our process-model which is called the ”Vorgehensmodell der EDVg” [2].

What we were expecting from this PIE was to gather experience by using all developed methods and techniques together under special attention of efficiency and effectiveness. A subgoal was to add tool support where applicable.

EDVg would be pleased to spread any results and lessons learned to the European Community as long as no confidential information concerning our baseline project has to be presented, we have no problems in publishing any of our results achieved through the experiment.

Company Background

Founded in 1963, our company (EDVg) has meanwhile branched out into a diversified conglomerate of different business units that provide information technology (IT) services, hardware and software products for a variety of customers. The 300 employees in the group gross some 60 mio ECU, which makes it one of the leading IT-companies in Austria. Apart from several specialized companies that have been formed with partners to service specific markets with their skills and products and in which EDVg holds different percentages of ownership, the IT-specialized business units are the following:

Software Development
Library Systems
Hospital- and Health-Care Systems
Membership-Organizations, solutions & services
Information Retrieval and Database-Services
EDVg-debis Systemhaus

Software Development

Originally almost exclusively focused on the development of applications software considerable emphasis has shifted to the planning, designing and implementing of integration projects involving a variety of standard software products on different systems platforms (e.g. workflow, groupware, internet or multimedia). Decades of experience and state-of-the-art-technology have contributed to the high level of quality that has come to be a trademark of EDVg's roughly one hundred engineers that work in this division.

Library Systems

As the leading provider of library systems in Austria, EDVg has begun to develop a new system, BIBOS:IV, that will gradually encompass all features that will have to support library institutions in the future. For large union-catalogues, as well as for single libraries, BIBOS:IV is characterized by an open client/server architecture, graphical user-interface, unix-base.

Hospital- and Health-Care Systems

For more than ten years EDVg has been active in that specialized segment of informatics. The result of a software development partnership with SAP is IS-H*MED, an R/3-based information solution for medical and nursing facilities within hospitals, that have decided to follow a SAP-strategy. Several references in Austria, Germany and one in Holland support the claim to quality and efficiency provided by this system.

Membership-Organizations, solutions & services

Information systems for large membership-organizations demand a high level of insight into the structures and the goal-setting of mainly non-profit organizations. Complex solutions for unions, an automobil club, political parties and a number of public and semi-official bodies have been designed, are being maintained and run by this specialized division, providing a usually quite heterogeneous group of users with a variety of highly integrated solution packages tailored to their specific organizational needs.

Information Retrieval and Database-Services

A large number of online and offline information sources is currently being used by a steadily growing clientele of EDVg. This business unit caters to the information needs of customers, private and public, by designing strategies to provide access to existing databases or by establishing adequate sources and offering data entry services, support and maintenance services inconnection with such a project. As of recent this business unit of EDVg also has formed a partnership with Fulcrum to distribute and support the Fulcrum products in their respective market segment.

EDVg-debis Systemhaus

A company jointly owned by EDVg and debis Systemhaus, a Daimler-Benz-Company, that is specialized in providing all system services that arise when planning, designing or operating an IT-infrastructure. Specifically, those services include network- and computing services, systems integration, facility management, backup and recovery services. Based on those skills differentiated IT-Consulting is available and offered on a project basis.

Software development is the core competence of EDVg. Therefore the competitiveness of our organization depends on the quality and productivity of the implementation of the software engineering process. One result of this situation is a strong need to improve the software testing process in terms of cost effectiveness and efficiency.

Starting Scenario

Strengths and weaknesses of our processes were investigated in the past via a BOOTSTRAP and a self assessment. The software development process is defined in all of its aspects (project management, process model, software development methods, change & defect management, configuration & version management, installation management, service management, human resource management, a.s.o.). Constructive as well as analytic quality assurance is an integral part of the operative project management during all phases of the software life cycle and is consequently implemented in practice.

Strengths:

The Quality Management System (QMS) of the Department for Software Development (UB04) is certified according to ISO9001 since February 1996. Since May 1993 a process model for software development and maintenance is mandatory for all

employees involved in software engineering. The process model is part of the QMS.
There also exists an organizational unit, responsible for the support, controlling and optimization of the Quality Management System resp. its processes and instruments.
In the current situation the challenge is to optimize the QMS to improve product quality, efficiency, cost effectiveness and flexibility of the development process in an environment, which becomes more complex technically as well as economically.
In many areas of the development process metrics are established to track quantitative information. Effort estimation is a controlled and well defined process based on Function Point Analysis, which is performed according to the rules of the International Function Point Users Group (IFPUG), published in their Counting Practices Manual.
EDVg is also one of the first users of CASE technology in Austria. As soon as there was a sufficient tool support available we integrated it into our process model. Over the years (we started with CASE in 1989) we developed a stable basis of models, which brings a lot of the anticipated advantages into our software development process. During the years we have never stopped supporting our development process with the appropriate tools especially by shifting from the traditional structural approach to world of object orientation. Today we are proud to say that we are one of the leading software developing companies in the client/server and in the inter-/intranet domain.

Weaknesses:

Testing was one of the identified problem areas. Test cases were documented, repeatable and therefore fulfilled the ISO9001 requirements, but efficiency, cost effectiveness, duration and effort of the testing process needed to be improved. The main facts to be addressed were:

Not enough attention is paid to the fact that testing is an intellectual and organizational challenging process in need of accurate planning on the one hand but a big potential for improving cost effectiveness on the other.

Testing types (module, integration and functional testing) are often mixed up.
Clear defined test environments, test cases and test data are often missing.
Tests are performed not systematically due to lack of time and resources.
Regression testing is insufficient because of poor tool support.
Configuration & version management is not well established for applications in the testing stages.
Metrics important for the testing process are sometimes missing (defects, testing coverage, a.s.o.).

Required Corrective Actions:

According to consciousness of our weaknesses we had to formalize our testing procedures in the same way as we already did when improving several other aspects of our software development process. We knew that this task couldn’t be performed in one step. Therefore we saw an evolutionary development from current status to the declared objectives.

The first step was the creation of the necessary awareness of all involved people (management and developers) for the importance of the topic. The next step was the introduction of a well defined method to guide the developers through the single phases of the testing process. One of the critical success factors in this stage was to provide support in terms of collaboration and interactively improvement of the given method as well as adding appropriate tool support for selected domains.

To convince not only the project workers but also the management, we finally have to prove that our introduced changes really brought an improvement to the whole software development process. This could only be done by measurement and comparison with other projects.

Plans and Expected Outcome

The fact that testing is one of the key processes in software development where the gap between theoretical concepts and practical implementation is wide, is well known in the software industry and described in the relevant literature [4],[5]. To show how this gap could be reduced was an implicit goal of our PIE.
An important prerequisite for evaluation the results under this given circumstances was to determine the current status of the whole process, to get a reference point for planned and achieved improvements during the experiment.
We hoped that the results of this experiment will give enough inspiration also for other software developing companies to look a little bit closer on their process and find some “hard” metrics. Testing in general and the influence on product quality and efficiency in the delivery could so become a cornerstone of everyone’s software development process.

Goals

There were two main motivation sources, both of which result in the need of performing a PIE concerning the testing process.

One came from the definition of the quality goals of EDVg’s ISO9001-certified Quality Management System, which postulates to increase customer satisfaction. As customer satisfaction is reached via a manifold of activities and has different reasons, one main trigger to achieve it is product quality. This quality again consists of many factors, but one of them is clearly visible and measurable, that is the number of defects detected after product delivery. This number can be reduced by optimizing the development especially the testing process, where EDVg has a great improvement potential according to the results of a BOOTSTRAP assessment, which was performed two years ago.

The second reason for starting activities to improve the testing process results from a software engineering point of view. The goal of having a mature organization, which is competitive due to the capability of producing high-quality software at reasonable costs, can only be achieved by analyzing, measuring and optimizing the software engineering process. As mentioned above, the testing process was already analyzed and found to be the candidate where an investment in optimization would be most urgent and where it will pay off quickly.

As competition gets tougher, it is also necessary to shorten development time, which is possible for instance by streamlining the testing process. Due to the different organizational responsibilities for requirements analysis and functional testing on the one hand and design, programming and module and integration testing on the other, the testing phase at EDVg was sometimes too long and too inefficient. Objectives of the process are poorly understood and so is tool support. One benefit of the PIE would be the improvement of all these influencing factors, which would result in a controlled and efficient process.

The productivity issue would also be positively addressed by the fact, that an optimized development and testing process requires less people in the maintenance phase and therefore more capacity can be put into the development of new projects.

We have tried to achieve both of the above mentioned goals within this experiment and give a detailed review about the results based on predefined metrics.

Subgoals

Based on our primary goals we have also tried to address two subgoals in this experiment. We have located a demand for an appropriate tool support within our Change and Defect Management Process (CDM). So what we have planned to do is an evaluation of the current situation on the test-tool market with a focus on regression and performance test tools.

A second aspect we have tried to cover is the whole administrative environment within the test process. Activities like capturing test cases, error tracking, documentation of tests and finally acceptance of the whole test case have to be documented in a simple and understandable way. Subsequently subgoal number two is to come up with an administrative solution for supporting the test process as well.

Implementation of Improvement Actions

The goal of this experiment was to implement all the intended improvement actions in a given baseline project. We therefore tried to find an innovative and representative development project on which we could show how well the improvements were achieved.

The baseline project was the " **OeKB Fulcrum-Application – Build Level 1"** (OeKB). The customer is the Österreichische Kontrollbank, Abteilung Bank u. Wirtschaftsinformation. This project was a customised development for the administration of various OeKB publications. The data are stored in an ORACLE Database with full text search capability provided by the Fulcrum Search Engine. The application front-end runs within standard Internet browsers (Netscape, Internet Explorer) and was developed using JAVA (JDK 1.1.6). The planned development effort for this project is 1342 PD.

OeKB is one of the first projects in EDVg to be implemented using 100% JAVA. All subsequent projects of this type should benefit from this experience.

The OeKB project was started on 4.7.1997 and in accordance with the schedule functional testing was started in March 1999 followed by acceptance testing in April 1999. Subsequently the application will be maintained under guarantee. PIE-ITC involvement commenced at the beginning of February during the preparation of test cases and the test environment for the functional testing. The synchronicity of the baseline project and PIE is therefore optimal.

A total of 4 developers are directly involved in the baseline project (1 project leader/analyst + 3 software designers/programmers). Temporary involvement of quality assurance, consultants, etc. has not been taken into consideration here.

Technical Impact

The following methods and tools which had either not been previously used in our other projects or only been partly used, were evaluated and/or introduced in the baseline project:

Test planning in early phases of the software life cycle.

Version 1 of the Test Plan is drawn up as an integral part of the project plan during the project conception. The Test Plan is continuously improved and extended as soon as the necessary information becomes available. We hope this will lead to a minimisation in resource bottlenecks during the project and thereby enable us to carry out the required test activities efficiently. This expectation has been met so far.

Systematic Drawing Up of Test Cases through Creation of Equivalent Classes and Border Values [6]

The number of possible combinations during the definition of test cases is easier to control through these methods while the ability to locate errors at the earliest possible stage is increased. We are aware that these methods are not a panacea for limiting the number of test cases and test efforts. However, they help in making a better informed and more transparent decision concerning the relationship between the desired test coverage and the amount of effort to be invested, thereby increasing the efficiency of testing.

In accordance with our PIE Project Plan, we evaluated tools for Functional Testing/Regression Testing, Module Testing and Performance Testing. Since there are now a vast number of such tools available, we used the Evaluations Reports from the Company OVUM [7] in the pre-selection process. We then carried out our own evaluations of those tools which met our pre-selection criteria.

We evaluated QACenter from Compuware and WinRunner from Mercury Interactive for Functional Testing/Regression Testing. We selected QACenter.

Through the use of QACenter we expect to achieve a significant increase in software quality and customer satisfaction as well as a significant reduction in total efforts, development efforts together with a reduction in the amount of resources tied up in maintenance activities.

The products QALoad (Compuware) and LoadRunner (Mercury Interactive) were selected from the OVUM Reports as possible Performance Test tools. Our expectations were a reduction both in total efforts and in the amount of resources tied up in maintenance activities as well as an increase in customer satisfaction.

Subsequent in-house evaluations were however stopped, since the licence costs for these tools (approx. 40.000 ECU) far exceeded the financial budget for the Experiment. We significantly underestimated these costs in our original PIE plan.

We evaluated Cantata++ from IPL as a Module Test Tool for C++. As a result of the use of this tool we expect an improvement in software quality and customer satisfaction. We place utmost importance on explicit Module Testing and in particular on the testing of programme critical modules.

As a result of the evaluation Cantata++ proved to be unstable in our development environment. We therefore decided not to use this product in the baseline project.

Organisation impact

Of course one of our main objectives in this experiment was to improve and therefore change the test processes. We do not really see any new roles or responsibilities that weren’t already defined in our ‘old’ testing process. What we try to achieve is to give the defined roles and responsibilities a more practical relevant meaning.

That means there are some organisational impacts caused by our new test process but basically this are enhancements and no new definitions. We would like to see that our new test process is easy to adapt in the different development projects on the one hand and is easy to control on the other and most of all shows the expected results.

Culture impact

Our organisation is certified to ISO9001. Our maturity is very high in comparison to the situation prior to certification and in comparison to other software development units. Our staff is accustomed to working in a structured fashion, to complying with necessary communications and documentation requirements and to working to continually improve QMS. The level of motivation is generally very high.

We have tried to involve our staff as widely and actively as possible. Prior to researching tools, we requested that the project leaders and operative management specify their requirements on the test tools. The response was high. The requirements were consolidated and used as the input for the tool research. After the test tools had been pre-selected, we asked the manufacturers to present their products to us. All relevant staff were invited to these presentations. On average about 20 employees took part in each presentation, which represents a percentage of 11,1% (from a total of 180 employees directly involved in software development).

In general the attitude to PIE is very positive. Our top management supports the project completely, the staff involved in the baseline project are highly motivated to contribute to the success of this project. There is no actual incentive for working in PIE. Part of the planned resources for the baseline project have however been explicitly reserved for PIE, so that no undesired extra workload arises for the employees directly involved in the project.

With those employees who are not directly involved we have experienced in some cases fear and even resistance on the one hand and yet on the other hand unrealistically high expectations.

One fear was that more control will be placed on developers if all tests and their results are documented. We tried to reduce this fear by pointing out the expected advantages both for the organisation as a whole as well as for the individual developers. As in the past, our aim is to increase the maturity of the whole organisation and not to place control mechanisms on individual employees. This has been emphasised both by PIE project management as well as by company management.

Some members of the operative management postulated an increase in overheads and in total costs as a consequence of longer development times for projects using the new tools and methods. In these cases we pointed out that PIE has an experimental character and the results of our experiment will show whether or not more effort has to be spent to succeed in our development projects using our new testing approach.

The unrealistic expectations concerned mainly the degree of support expected from test tools. We pointed out that there is no tool available which can replace the intellectual input of the developer during testing and above all in the drawing up of test cases and then find all errors automatically at the press of a button and then declare the software to be bug free. We also pointed out that the drawing up and maintenance of test cases involves a large amount of effort which is not recovered until the test is re-used during regression testing. The definition of criteria as to when the use of a test tool makes sense or not is an integral part of the optimisation of the existing test concept.

Measurement of Results

Measurement is an essential part of the whole experiment and the only way to show if and how the different tasks within the experiment effected the overall performance. We have raised the necessary data out of the different development projects during the last years which is part of our Quantitative Quality Management and now we try to compare them with the figures we have gathered during our experiment

Basic metrics measured and analyzed are:

Function Points

Function Points are measured according to the rules published by the International Function Point Users Group (IFPUG), where EDVg is a member of.

Defects

Defects are defined, measured and tracked according to the Change- & Defect Management process, which is also part of the QMS.

Time and Effort

Time and Effort are recorded via the Internal Administration System, which can be related to a project management tool. Costs are highly related to effort and therefore tracked via effort recording.

The approach we have chosen here is called Goal Question Metric see p.60 in [10]. This is a strictly top-down approach where first the goals are defined then you are looking for the questions which will lead you to this goal and finally based on this question the appropriate metrics are derived.

In our case based on the basic metrics, the relevant metrics for the baseline project according to the PIE objectives are defined as follows:

Immediate (i.e. after completion of the baseline project) available metrics:

TESTING EFFORT 1:	total testing effort per function point
PROCESS PRODUCTIVITY 1:	total development effort per function point
PROCESS PRODUCTIVITY 2:	total development time per function point
TESTING EFFICIENCY 1:	number of defects per function point found during testing
TESTING EFFICIENCY 2:	number of defects found during testing in relation to total testing effort
PROCESS QUALITY 1:	total testing effort in relation to total development effort

Long term available metrics:

PRODUCT QUALITY 1:	number of defects per function point found up to 6 months after delivery
PROCESS QUALITY 2:	total development effort in relation to total maintenance effort (6 months after delivery)
PROCESS QUALITY 3:	number of defects found before delivery in relation to number of defects found 6 months after delivery

The above metrics should be compared with the current values and values taken from literature and/or benchmarks, to which EDVg contributed their data already in the past, e.g. Howard Rubin’s Worldwide Benchmark Project 1995 [8], IFPUG Benchmarking Data Base 1994 [9], etc.

Long term metrics become available at least 6 month after finishing the base line project.We are planning to deliver an extra long term experience report to the community and to all interested companies with whom we got in touch during the experiment.

The final results of our experiment comparing the data we derived from our baseline project compared with our standard figures are shown in the following table:

Measurement attribute	**Pre-experiment numbers**	**Plan**	**Achieve-ments**
TESTING EFFORT (TE) total testing effort per function point	0.91Ph/FP	0,83 Ph/FP	0,86 Ph/FP
PROCESS PRODUCTIVITY 1 (PP1): total development effort per function point	13,9 FP/PM	15,3 FP/PM	15,1 FP/PM
PROCESS PRODUCTIVITY 2 (PP2): total development time per function point	19,7 FP/M	21,7 FP/M	20,1 FP/M
TESTING EFFICIENCY 1 (TE1):number of defects per function point found during testing	0,1 Def/FP	0,13 Def/FP	0,22 Def/FP
TESTING EFFICIENCY 2 (TE2): Number of defects found during testing in relation to total testing effort	0,19 Def/Ph	0,25 Def/Ph	0,27 Def/Ph
PROCESS QUALITY 1: (PQ1) total testing effort in relation to total development effort	9,14 %	7,95 %	8,63 %
PRODUCT QUALITY 1: (PQ2) number of defects per function point found up to 6 months after delivery	0,037 Def/FP	0,032 Def/FP	N/A
PROCESS QUALITY 2: (PRQ1) total development effort in relation to total maintenance effort	31 %	26,5 %	N/A
PROCESS QUALITY 3: (PRQ2) number of defects found before delivery in relation to number of defects found 6 months after delivery	N/A	N/A	6.67:1

Presenting these figures you have to consider that this project is just an experiment and therefore a snapshot regarding the statistical relevance. We would like to point out that we have achieved a positive trend in all of the above mentioned metrics even if we did not reach our expectation in all categories.

You can find some metrics which definitely show a significant improvement (TE1)and other which can be stated as statistically irrelevant (PP2). What we would like to do in the future to prove these results and emphasize on those metric where we can see the biggest potential for improvement and see the others as a more or less welcome side-effect.

References

[1] ADV, Software Quality the Way to Excellence, Proceedings of the sixth European Conference on Software Quality, 1999

[2] Elektronische Datenverabeitungs Ges.m.b.H, Online Vorgehensmodel in IPF and Win-Help Format, 1991-1999

[3] Myers, Glenford J., Methodisches Testen von Programmen, R.Oldenbourg, München/Wien, 1991

[4] Humphrey, Watts S., Managing the Software Process, Addison Wesley, 1991

[5] Ince, Darrel, Software Quality and Reliability, Tools and Methods, Chapmann & Hall, 1991

[6] Poston, Robert M., Automating Specification-Based Software Testing, IEEE Computer Society 1996

[7] Ovum Evaluates: Software Testing Tools, Editor: Graham Titterington, Manager: Eric Woods, Continuously updated

[8] Rubin, Howard A., Worldwide Benchmark Project, Rubins Systems Inc, 1995

[9] IFPUG, ISBSG Benchmarking Repository Report, periodical report issued evry 9 month by the IFPUG

[10] Fenton, Ne: Software Metrics – A Rigorous Approach., Chapman & Hall, London, 1991

CV: Martin Prieler (Senior Consultant)

Professional experience:

various projects in the IT-domain since 1987

participation in different medium to large development projects (as analyst, quality agent, consultant and project manager)

methodology support and adviser (processing models + project management, structured and object oriented methodologies)

tool support (project management-, test- and CASE Tools)

consulting in IT-specific topics with an emphasis in:

➔ implementation of IT-strategies and solutions

➔ design of feasibility studies

➔ project supporting consultation and quality assurance

➔ answering of several huge invitations to bid

experience in following branches of industry: banking, insurance, public administration, other service oriented industries)

Education:

Grammar school (1985)

Computer Science on Technical University of Vienna (1991)

further professional education the following domains:

➔ software development methodologies

➔ project management

➔ cost and effort estimation of IT-Projects

➔ IT-related sales and marketing

additional economical qualification

Project experience (past three years)

Österreichishe Kontrollbank (OeKB)
Intranet JAVA Based development project

Österreichische National Bank (OeNB)
feasibility study for project migration

BIBOS IV Library System
OO methodology

MVZV
methodology and tool support, quality assurance

European Multimedia Schoolnet (EUN)
project management

Bundesländer Versicherung (BARC Group)
consulting and improving of development process

Company description

Comany:

Elektronische Datenverarbeitungs
Gesellschaft m.b.H.
Hofmühlgasse 3-5
1060 VIENNA
Austria

Contact person:

Dipl.-Ing. Martin PRIELER
tel: +43.1.59907.1438
fax: +43.1.59907.1363
email: martin_prieler@edvg.co.at
web: http://www.edvg.co.at/

Practical Measurements for Reengineering the Software Testing Process

P. Pongas
SINGULAR
29, Alexandras Ave., 114 73, Athens, Greece
tel. +30 1 647 9600, fax +30 1 646 9534, ppong@singular.gr

S. A. Frangos
ABS Group of Companies
6 Skouze str., 185 36, Piraeus, Greece
tel. +30 1 429-3804, fax +30 1 429-3809, cymentor@hol.gr

Abstract

This paper focuses on software testing and the measurements which allow for the quantitative evaluation of this critical software development process. Innovative software producing units are committed to continuously improving both the software development process and the software product in order to remain competitive in today's global community. Software product quality and software process improvement commence with addressing the testing process in a quantitative manner. The continuous monitoring of the testing process allows for establishing an adequate level of confidence for the release of software products and for the quantification of software risks, elements which traditionally have plagued the software industry.

The identification and removal of software defects constitutes the basis of the software testing process a fact which inevitably places increased emphasis on defect related software measurements. Defect Distribution, Defect Density and Defect Type metrics allow for quantifying the quality of software modules, while Defect Age, Defect Detection Rates and Defect Response Time metrics allow for pinpointing software inspection and testing process shortcomings. Code coverage and testing effort measurements complement the defect metrics and provide additional software product as well as process quality indicators. The paper concludes with the presentation of the application of testing metrics in industry with a focus on SINGULAR's ESSI STAMP process improvement experiment.

Keywords

Defects, Testing, Metrics

1 SOFTWARE PROCESS AND PRODUCT IMPROVEMENT

"Neither have you ever contemplated what kind of people are the Athenians with which you will have to compete and to what degree they are different from you. They are innovative and progressive and quick in the implementation of their plans, while you confine yourselves to what you already have, without coming up with something new, and when you act, you do not even cover the absolute minimum." Thoucydides

Athens, during the golden era of Pericles, evolved as a leader in the entire known world and accomplished achievements which have yet to be surpassed. The life style, the achievements and most importantly, the principles which governed the conduct of the ancient Athenians constitute good background material to any professional seeking to find answers to many of the most complex issues which modern society is confronted with. In the above quote from Thoucydides, who many consider as the top strategist of all times, the secret of the Athenian success, versus arch-rival Sparta, was product innovation, continuous improvement and growth as well as the ability to rapidly implement planned actions. Thus, the strategy which modern software producing units must adopt is well-known, however, what remains to be planned in corporate brain-storming sessions is the tactical plan for achieving the strategy.

Global markets have increased competition dramatically which has resulted in the need for software development firms to produce at a lower cost, with higher quality and within shorter time frames. The focus must clearly be on the customer and the objective must not simply be to satisfy, but to delight. This can only be accomplished by providing the right system and executing the pertinent project(s) in the right way. Providing the right system translates into providing a system to the customer which reflects both stated and implied requirements. Doing things the right way can be achieved by validating and verifying requirements, for both external and internal customers, during the entire project life-cycle. Figure 1 depicts the customer satisfaction matrix [Lowell 1992].

Figure 1 - Customer Satisfaction Matrix

	Right Way	Wrong Way
Right System	Delighted Customer	Satisfied Customer
Wrong System	Dissatisfied Customer	Angry Customer

The demand for new services and products adds another dimension to the already challenging strategy. Customers request variations to software products so as to meet changing technology advances and their specific needs. Therefore, the customization of software products must be accomplished both rapidly and on a large scale in order to allow the products to become, or to remain, competitive.

All quality gurus advocate that the quality of the offered services and products is dependent on the respective software development process. The journey to achieving a well-defined software development process, capable of supporting customized products, is not a simple task, therefore, an incremental approach is required.

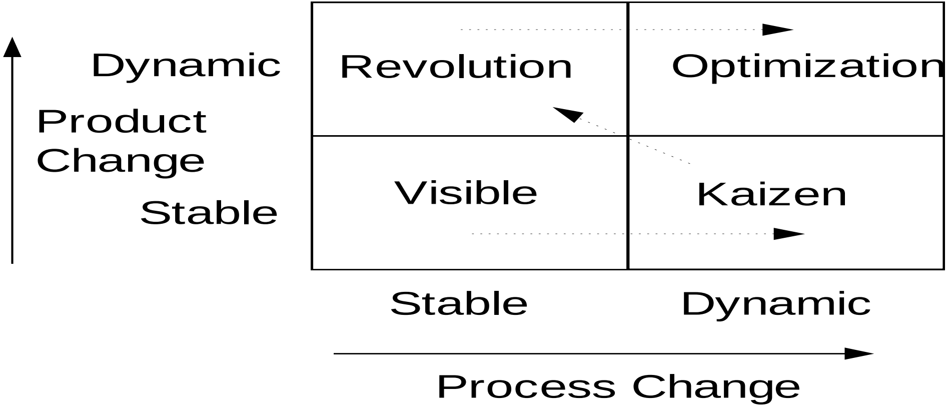
The initial phase entails making the software development process visible. This is accomplished by recording the actual activities that are required to produce software products. Many software firms use formal methods or process modeling tools to accomplish this first and crucial step. Once the development process is described and stable, it is important to institutionalize it across all projects so as to make the software development process repeatable. This will enable similar projects to be executed in a similar manner.

The second phase requires going one step further, to process control. At the tactical level, measurements are taken throughout the software development project so as to allow project managers to base their decisions on actual project data. At the strategic level, collective data from all completed projects are analyzed so as to review the software development process. Thus, opportunities for improvement are continuously identified as process improvement is a never ending evolutionary process (kaizen). Once a well-defined and effective development process is in place, the orientation can shift to the product.

The basic challenge for today’s software firms is to provide clients with customized products and to provide them fast. For this to be accomplished, small and flexible organizational units and reusable software components must be set up in a loosely coupled network structure. The project manager in such an organization, who can be seen as network coordinator, makes best use of the available resources (i.e. engineers and software components) on a project by project basis. The ability to mass customize means that software development is not only evolutionary, but revolutionary as well.

The final stage is process optimization, whereby the process is adjusted in accordance to the extracted project measurements “on-line” and customized products are offered. Figure 2 outlines the different levels of process maturity.

Figure 2 - Process Maturity Navigation Matrix



2 SOFTWARE TESTING PROCESS REENGINEERING

The continuous improvement of the software development process commences with handling it’s weakest link, the testing process. If one accepts that the strength of a chain is equal to the strength of it’s weakest link, the importance of reengineering the testing process is evident. Prior to reenginnering the software testing process, the testing objectives must be established so that the software testing process is in the position to correspond to clearly defined goals. Hetzel [Hetzel 1988] lists the following list of practitioner objectives regarding software testing.

Checking programs against specifications
Finding bugs in programs
Determining user acceptability
Insuring that a system is ready for use
Gaining confidence that it works
Showing that a system performs correctly
Demonstrating that errors are not present
Understanding the limits of performance
Learning what a system is not able to do
Evaluating the capabilities of a system
Verifying documentation
Convincing oneself that the job is finished

In order to meet such testing objectives, most software producing units that have reengineered their testing process have more or less in some STEP-like testing process. The STEP testing process can be summarized by the following major testing activities [Hetzel 1988] :

PLANNING
PLAN the general approach
DETERMINE testing objectives
REFINE the general plan
ACQUISITION
DESIGN the tests
IMPLEMENT the tests
MEASUREMENT
EXECUTE the tests
CHECK termination
EVALUATE results

The reengineering of the testing process must be accompanied by practical measurements which will allow for continuously monitoring the process and for assessing the quality of the software products.

3 PRACTICAL TESTING MEASUREMENTS

The software testing process requires practical measurements for the quantification of all software testing phases. Starting from the planning and acquisition phases of the software testing process, one first has to recognise that current software development practices do not segregate coding effort from unit testing, but rather, coding and unit testing are seen as one software life-cycle phase. Therefore, the duration of the testing phase basically covers test planning effort, integration and system testing. The following measurements apply to software development effort distribution [Rubin 1995] :

Analysis	16%
Design	17%
Code/Unit Test	34%

System/Integration Test	18%
Documentation	8%
Implementation/Install	7%

Regarding software development distribution effort, Grady [Grady 1992] reports the following breakdown from the analysis of 125 Hewlett-Packard projects :

Specification/Requirements	18%
Design	19%
Code/Unit Test	34%
System/Integration Test	29%

Project managers are thus encouraged to plan enough time for the testing phase and to envisage approximately 20% of total software development effort for testing. This is very crucial as many software development projects run into serious problems being that software delivery dates are rigidly defined in the pertinent contracts, while the respective analysis and design delivery dates are consistently overrun. Many project managers in order to overcome such situations truncate the testing effort in order to meet contractual requirements for the project delivery dates. Unfortunately, the author does not know of any project which benefited in the long run from such testing effort truncations.

The project manager must also be in a position to predict the number of test cases required so as to test adequately the developed software. Capers Jones [Capers 1996] has done extensive research for such measurements and has concluded that the relationship which governs test cases and function points is the following :

Number of Test Cases = (Function Points)$^{1.2}$

Capers Jones [Caper 1994] has also qauntified the size of the test plan in pages per function point. The average number of pages created per function point for software project is 0,25 for system software, 0,10 for MIS software, 0,55 for military software and 0,25 of commercial software.

To conclude, for the quantification of the planning and acquisition phases of the testing process, the percentage of testing effort with respect to the entire software development effort is known and the number of test cases which must be generated for the adequate testing of the software product is more or less predictable. For software producing units which are not using function points, but Source Lines of Code (SLOC), the relationship between function points and SLOC is approximately 1 Function Point per 100 SLOC depending on selected programming language.

The measurement phase of the testing phase is perhaps the most interesting and certainly the most significant. As software code is the single most important deliverable in any software development project, incereased emphasis must be placed on ensuring that the right information system is being provided (validation) and that it is developed correctly (verification). Two are the basic principles which govern the entire measurement phase of the software testing process. They are :

Pareto 20% of the code causes 80% of the problems
80% of the transactions traverse 20% of the code

Code Coverage The reliability of software is dependent on the traversed tested code

In order to apply the pareto principle, project data must include details pertaining to the detected defects. Invariably, an analysis of the detected defects allows for a good understanding of the information system strong and weak links. Defect related measurements include the following [Grady 1992], [Hetzel 1993] :

Defect Mode : Defect classified as either “missing”, “unclear”, “wrong”, “changed” or “better way”
Defect Origin : Defects recorded per System Configuration Item (SCI). SCIs include code units, technical and user documentation, deliverables, etc.
Defect Density : Defects per software size measured in either Source Lines of Code (SLOC) or Function Points
Defect Age : The time of introduction of defect to time of detection. To compute the measurement one assigns a number to each software development life-cycle phase and calculates difference between detection phase with introduction phase (i.e. “analysis” can be assigned a 1, “design” a 2, “coding” a 3, etc. for computation of defect age).

The analysis of defect related measurements is based on comparisons of retrieved values with inter-company and international average defect values. Moreover, cross-examinations of defect measurements allows for both validation of results as well as for drawing meaningful conclusions about the effectiveness of the testing process. The following table summarizes the usage of defects and possible testing process shortcomings based on cross-examination of retrieved defect values with average industry values.

Metric	Analysis of Testing/Development Process
Defect Mode	“Missing” defects is an indication that either not enough customer research took place or that “requirements” did not reach implementation phase. “Unclear” and “Wrong” defects usually means not enough time spent on analysis & design inspections, specification language is vague or customer was not in a position to specify business requirements. “Better way” defects usually reflect lack of design inspections or bad coding practices, while “Changed” defects are an indication of high requirements creep or inappropriate understanding of hardware / system software platform.
Defect Density	Excellent means of quantifying software product quality as well as to assess the effectiveness of the testing process. Many industry averages exist, most of which are in the area of 10-20 defects per one thousand lines of code (KLOC) during system testing, while prior to and including system testing values are up to 200 defects per KLOC [Humphrey 1996]. Rubin reports defect rates from a world-wide survey per industry sector with Aerospace at 4.6 defects / KLOC, Financial at 3.1 defects / KLOC and System Software 2.0 / KLOC [Rubin 1995]. Rubin also reports 0.9 defects per function point as a world-wide industry average, while Capers Jones reports 1.75 coding defects per function point [Capers 1995] [Rubin 1995]. Defect densities should drop ten-fold between testing and maintenance phases [Grady 1992]. If such trends are not realized between prerelease and postrelease of software, then this is an indication that the testing process is not effective. This metric should also be used in parallel with the percentage of effort spent on testing. Limited testing time allocated will result in few defects detected prerelease and many detected postrelease.
Defect Origin	Probably the most useful metric for evaluating information system product quality. By depicting with bar charts all defect densities per SCI, the project manager can pinpoint the weakest links in the software product. Obviously, weakest links must be retested and perhaps, even redesigned. Defects tend to concentrate on certain portions of the entire information system (pareto principle). Once the defective modules are identified, they need to be test drilled to no end.
Defect Age	A good “barometer” of effectiveness of in-process inspections. High averages for defect age are a clear indication that analysis and / or design inspections are not effective.

Software reliability, the most important constituent of software quality, is a function of code coverage. Therefore, in order to put a handle on software reliability problems, the coders must be in a position to assess code coverage. Typical testing without measuring code coverage only exercizes around 55% of the code, while with the use of code coverage instrumentation, this can be raised to at least 80% without excessive additional effort [Grady 1992]. The insertion of numbered checkpoints in the code (i.e. “Function X, Checkpoint N” with fprintf statements) is an alternative to coders, if automated dynamic analysis is not available through the use of software testing tools.

The analysis of code coverage during system testing will also allow for identifying which code segments get exercized the most during the execution of business transactions. By identifying the code segments which get exercized the most, the testers can focus on most heavily traversed statements while designing and executing tests. Obviously, time limitations do not allow for complete code coverage and testers must maximize testing benefits versus the time allocated to them for testing.

Perhaps the most important testing measurement is the one which will provide an indication concerning the readiness for the software code to be released. Most project managers which want to release code based on testing measurements generate graphs of cummulative number of defects detected per some meaningful unit of time (i.e.hours, days or weeks depending on module size). The slope of the curve will steadily approach zero as the testing process concludes and the software code can be released. Such graphs, per each code module, are an important project management tool for monitoring the maturity of the respective modules regarding defect detection and correction.

4 STAMP - PRACTICAL TEST MEASUREMENTS AT SINGULAR

The technical impact of STAMP to Singular's software development process has already made an extremely positive impact. Testing related principles and measurements are now understood for the first time with such detail and it is expected that all software development projects within the company will benefit from the dissemination of the acquired knowledge to non-STAMP Singular software developers.

The generation of the STAMP measurement plan was a gigantic step forward for the organization. Testing was previously seen as an art, where the outcome of the software testing process was primarily dependent on which software engineer was assigned the task of product testing. Now, the testing process is well-understood and the collected testing related measurements allow for establishing reliable **entry / exit criteria** from each software testing phase. More specifically, the following measurements were identified and introduced as part of the experiment :

Defects - Detected defects classified (logic, data handling, computation, etc.) and registered as either missing, unclear, wrong, changed or better way for each baseline project software module tested. The defect densities for each software module will be calculated as the ratio of defects per software size (SLOC).
Test coverage - Percentage of items (requirements, test features, programs, code statements and branches) covered during testing.
Product Reliability - Failure rate and Mean Time between Failures (MTBF) during the test period and after release.
Product Perceptions - Direct measures of perceived effectiveness
Test Analysis - Analysis and study of tests to measure testing effectiveness.

A major technical objective set by management in the planning phase of the PIE was to be in a position to pass judgement on the software product's reliability prior to release to the customer. It is believed that the above listed measurements will allow for the quantification of the testing process to the extent required, and certainly will constitute reliable release criteria for the baseline project software product outcome. STAMP deliverable D0.3 entitled "STAMP Measuerement Plan" elaborates in detail on the identification and definition of all testing related measurements to be used in STAMP PIE.

Another major technical objective was to identify an appropriate testing tool, which is congruent with the overall software development process. During the Selection procedure of the testing tool, the purpose was defined and the selection criteria were identified and weighted. Based upon the evaluation results and the application of selection criteria, a decision was made, about the testing tool. The selection and the evaluation processes of the testing Tool interacted with one another. On the basis of the evaluation results obtained, the goals of the selection process and/or the selection criteria and their weights sometimes required modification and were fed back into the evaluation process. The following evaluation criteria were used :

Testing life-cycle phases supported
Test Planning
Test Design and Creation
Recording
Tool Customization
Script Language
Test Execution
Playback
Verification, including Static Analysis
Debugger
Code Coverage (Dynamic Analysis)
Report and Analysis
Report Capabilities
Report Customization
Charting Capabilities
Problem Tracking
Defect Logging
Workflow Tracking
Documentation
Testing tool Repository
RDBMS which stores the repository
Capability to copy test procedures from one repository of the testing tool to another repository of the same testing tool
Simultaneously multi-user access
Configuration
Development Environment Supported (e.g. Oracle Developer/2000, Delphi, Visual Basic, PowerBuilder)
Operating System (i.e.Windows 95, NT, UNIX)
Application-under-test type (i.e. The applications-under-test could be Windows client/server applications or character-based UNIX applications, or Web-base application)
Ease of Installation and Learning
Installation Process
Tutorial
On-line Help

The following testing tools were evaluated, prior to selecting **Rational's SQA**:
Rational's SQA Suite TeamTest Edition,
Compuware's QA Run - QADirector - QATrack,
Vermont Creative Software's Vermont High Test Plus
Segue Software's QA Partner

STAMP deliverable D3.1 entitled "Testing Tool Report" elaborates in detail on the testing tool evaluation process and on the results generated from evaluation of the above listed four testing tools.

5 CONCLUSIONS

Successful software development units worldwide have incorporated within their business strategy the continuous improvement of their development process and of their product line. Such strategic objectives require the identification of process elements which have a significant impact on product quality. Software testing is a process area which traditionally needs improvement and the metrics presented in this paper allow for the quantification of both the product quality and of the respective software testing process.

Software firms which have used all or even some of the presented metrics achieved significant improvements. This is achieved due to the fact that line management has visibility to the development process and decisions are not made based on intuition alone, but, with the sound interpretation of the available software testing metrics. As with any other process improvement, management needs to establish a customized action plan and must be in a position to communicate the plan to all interested parties, including senior management, so as to achieve the required “buy-in”.

6 REFERENCES

Capers, J. June 1994. Revitalizing Software Project Management, American Programmer.

Capers, J. 1996. Applied software measurement, McGraw-Hill.

Frangos, S.A. 1995. Implementing a quality management system using an incremental approach, SQM 95 Proceedings.

Grady, R.. 1992. Practical software metrics for project management and process improvement, Prentice-Hall, Inc.

Hetzel, B. 1988. The complete guide to software testing, QED Information Sciences, Inc.

Hetzel, B. 1993. Making software measurement work : building an effective program, QED Information Sciences, Inc.

Humphrey, W. 1996. A Discipline for Software Engineering, SEI Series for Software Engineering.

Lowell, A. 1992. Improving software quality : An insider’s guide to TQM, Wiley.

Rubin, H. 1995. Worldwide benchmark project report, Rubin Systems Inc.

Venizelos, E. 1937. Thoucydides History (translation from Thucydides), Georgiades Publications.

7 VITAE

Panayiotis Pongas is the project manager for the the PIE experiment for Singular S.A. He received a Msc degree in Electrical Engineering from National Technical University of Athens, GREECE. In the recent past, he was employed as the project manager of Singular S.A, a leading Greek software firm. He was employed in various Projects as well as management roles worldwide and has acquired an expertise in quality engineering, methodologies. He is the author of several conference papers and financial journal articles on software quality.

Stelios Frangos is the business development manager Europe for the USA based ABS Group of Companies. He received a BS degree in Chemical Engineering from Rutgers

(1983), USA, and a MS degree in Computer Science from NJIT (1986), USA. In the recent past, he was employed as the general manager of the Advanced Training Center Bull, a leading Greek industrial training firm. He is the principle designer of ISO 9001 certified QMSs for leading IT firms in Greece. He was employed in various software engineering as well as management roles worldwide and has acquired an expertise in quality engineering, methodologies, CASE tools, metrics and project management. He has delivered many seminars on software quality as well as on software project management. He is the author of several conference papers and financial journal articles on software quality. His latest interests include software QMSs, environmental management systems and quality certifications.

Session 3
SPI and Re-Use

Chairman

Yingxu Wang

IVF, Gothenburg, Sweden

Finding a Practical Approach to Organised Reuse

Roar Tørlen
PROVIDA ASA, Ålesund

Ulf J Krystad
PROVIDA ASA, Oslo

Introduction

For a company like Provida, being a small/ medium sized enterprise operating in a high-competitive international market, it is important to offer flexible and modularised systems within short notice in order to stay competitive. We believe that our process improvement steps focusing on *reuse* will contribute substantially to this.

Through our participation in a PIE (Process Improvement Experiment) project we have experienced the need for addressing certain key aspects; *roles and procedures regarding reuse in practice to be defined and properly implemented into the organisation*; *the needs for education and training to be focused when making such a paradigm shift; the definition of the requirements for a repository from a reuse point of view to take place.*

When introducing organised reuse one should take special precautions to avoid some typical traps. Some of them being underestimation of the need for consensus in the organisation for the change, not having a plan for implementing the new roles and procedures into the standard development environment, and underestimation of the need for some constancy in the surrounding environment while performing the change.

Even if this is a long-term investment and the payoff is expected to materialise in coming projects, we already now see clear evidence that organised reuse gives a shorter time to market.

PROVIDA ASA, Business and Products

Provida is a software house for the banking industry, in the Esprit terminology classified as an SME (Small & Medium large Enterprise). The company is divided into 5 divisions, and the PIE project was run in the Retail & Corporate Division.

Provida Retail Division develops large systems for retail and corporate banking, including Customer Information, Loan, Deposits, Corporate Accounts, and Card Management Systems, marketed in Europe, South America and Australia, under the umbrella ProRetail. ProRetail is marketed as an international basic version, a version for each country and a customer-specific version. A new delivery of the system was always based on a copy of the best-suited existing version. Changes to this version were done according to national or customer-specific requirements, creating a complete new version of the system, which was maintained separately. We realised that this practice would increase maintenance costs dramatically on a long-term basis. Furthermore, we experienced that the development costs for each delivery were too high and that we were not able to take full advantage of previous development in a new delivery. This problem arose since the division was not properly organised to address the aspects of reuse.

Starting Scenario

Experiment context

For a company like Provida, being a small and medium sized enterprise (SME) operating in a high-competitive international market, it is important to offer flexible and modularised systems within short notice in order to stay competitive.
One way of improving our development process was to introduce and test out *organised* reuse, using the method set forth by the ESPRIT project REBOOT (project no: 7808). With financial aid from the EUROPEAN COMMISSION, DGIII-F3, the ESSI program, experiments were carried out on development projects: *organised* reuse were introduced as the technology being used, part of extra costs incurred by the switch in technology being supported and financed by the Commission. The PIE project was named REPRO (REuse PRocess and Organisation improvement experiment).
Following REBOOT, it was planned that REPRO would work *directly* with two baseline projects, focusing on *development for reuse* and *development with reuse* respectively. We wanted to show that the introduction of *new roles*, *procedures* and *tools* in the first baseline project would contribute to creation of libraries of reusable components that could be *reused* in the second baseline project. Furthermore, we wanted to show that this strategy was cost-efficient by using metrics and measurements considering all aspects of the development process. REPRO's role in the baseline projects would be to direct and assist in the adaptation of our new procedures, as well as performing quality control on the results.

The Experiment should introduce organised reuse by focusing on the following key reuse areas:

1. Organisation and Project management
2. Development *for*, respectively *with* reuse
3. Repository management
4. Metrics and measurements

To do this, certain improvement steps were identified:

- The introduction of specific organisational roles, responsible for reuse, that can improve the degree of re-usability in our projects. The roles, taken from the Reuse Maturity Model, are typically *Domain expert, Repository Manager, Component co-ordinator, Component expert, Development for reuse expert* and *Reuse co-ordinator.* One person might cover several of these roles.
- The definition and use of required activities for reuse in all relevant phases of our development processes as well as the incorporation of these activities as amendments to current procedures for project management.
- The introduction of a new tool for repository, that covers the functionality of both our old repository and old change management system. We assume that the use of the appropriate repository, is a major contributor to better reuse practices.
- The introduction of Object Oriented design principles in the building of component libraries. Take special design considerations to deal with generalisation and specialisation problems.
- Show that our COBOL 85 implementations could be converted to Object Oriented COBOL, and show that this improves productivity and reuse.
- Use metrics and measurements, to show the cost-benefit of the above mentioned improvement steps.

Documentation is also relevant. It is implicitly understood that all information stored in the repository could be output as system documentation according to standards.

Status before the experiment

We assessed and analysed our practices according to the ESSI questionnaire, the Capability Maturity Model (CMM) and the Reuse Maturity Model (RMM) – see glossary for references. With respect to CMM, the company was typically at level 2 and did also satisfy most of the requirements of level 3. The ESSI questionnaire showed that we had good practices for standards and procedures, for control of the Development Process and for Tools and Technology. In fact, we had procedures and checklists for all phases of the development process.

To be more specific; Requirements were specified, managed, and maintained in Lotus Notes databases. Projects were defined using standard templates in Process Engineer, and scheduled in Microsoft Project. All activities were tracked, and earned value calculated and reported for all deliverables. Quality control was performed in the project for each deliverable and external formal review was performed for all major phases. All units were under configuration management control from unit test and onwards. All future changes were handled according to formal procedures.

For development we used Microfocus COBOL Workbench for programming, Datamanager from MSP as the repository and CCC from Softool as the change and configuration control system. Datamanager and CCC were not integrated, and both tools run on a mainframe with a character-based interface. We clearly saw that the development process suffered under this.

Reuse practices

Analysing our practices for reuse, using RMM, we found that we were somewhere between level 1 and 2. We did have reuse for product plans, contracts, specification and

documentation, but seemed to have little organised reuse in the development process where formal procedures were missing and roles not defined.
We had grouped our development department according to business domains of our products. However, the personnel were owned by a single project, where the project tended to focus on local goals instead of corporate product strategies. Personnel, solely responsible for reuse, did not take part in the projects. Hence, reuse-specific activities were not incorporated in project plans and reuse practices were at an ad-hoc level, dependent on the individuals in the projects.

Conclusion

According to CMM, we found that we typically had a defined level for our development process, almost satisfying level 3 requirements. However, our poor reuse practices had made it difficult to have a common basis with a single source for our standard versions of the products. According to RMM, we were at level 1 or 2.

Plans and Expected Outcome

Project objectives

The overall goal was to improve our development process through *organised* reuse, and reach a higher maturity using RMM. The main objectives were to *implement development procedures and roles* focusing on reuse, to *build libraries of reusable components* and hence; *improve quality*, *increase productivity* and *reduce time-to-market.*

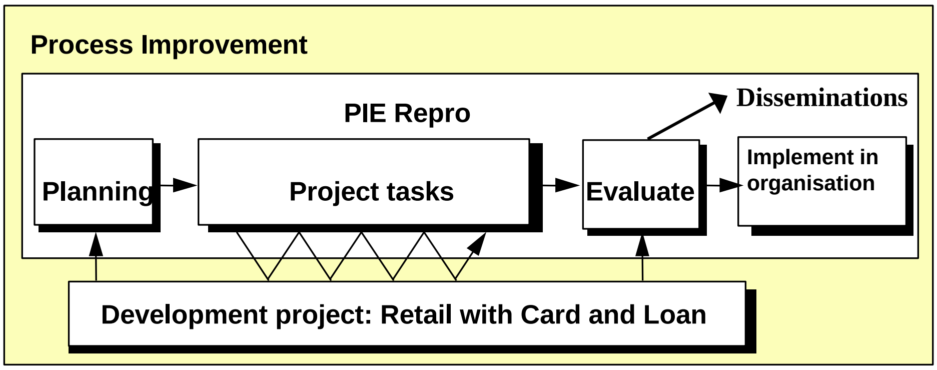

Process Improvement
PIE Repro
Disseminations
Planning
Project tasks
Evaluate
Implement in organisation
Development project: Retail with Card and Loan

Figure KryTor.1: Improvement interaction with baseline project

The yardsticks for how we wanted to measure any results are listed below. We wanted to measure an *overall* process improvement on the basis of a reuse assessment report in the preparation phase and in the evaluation phase of the project. The assessments were to be based on RMM and should describe our reuse practices before and after the PIE (Process Improvement Experiment).

Implement development procedures and roles focusing on reuse

The definition of new procedures and the use of specific roles, should be a result from the work done with regard to *reuse strategy* and *role and procedure description.*

Build libraries of reusable components

This could be measured by the number of components built. The actual *reusability* was to be measured in the "development-with-reuse" phase to see how often a component is reused and what changes are needed to use a component in another context. The time-effort spent to tailor existing components to meet new requirements, together with the time spent to build reusable components and the effort saved in *reusing* them, should be the major parameters in the cost-benefit analysis measuring the *value* of the libraries.

Increase productivity and reduce time-to-market

Increased productivity should be measured as reduced man-hours used to accomplish certain functionality. Any results will first be visible in the "development-with-reuse" phase. Reduced time-to-market is related to increased productivity and can be measured by the number of requirements met by the use of existing components. We wanted to use the Repository actively during analysis and design in the baseline projects to search for components that meet specific customer criteria; outlined in a *Functionality evaluation report.*

Improve quality

We wanted to use the Factor-Criteria-Metric model to measure the factors reliability, maintainability and reusability. Since we can never measure quality *exactly*, and since we need some experience with maintenance over time; we did not expect to have good measurements on quality until the later phases of the project.

Baseline project context

The selected baseline project was named **Retail with Card and Loan**. It was an internal project of strategically high importance for our division.
Originally, we wanted to choose a part of the project as a basis for experiment – called PIE domain. It was more realistic (and practical) to use the new methodology for a complete project. Hence, the PIE domain was the whole project
Retail with Card and Loan. ProRetail is a banking system running on an MVS mainframe. The strategic target sub-systems are CICS and DB2. At this point two products, ProCis and ProDeposits had been developed to that platform. The other products in the ProRetail family, ProCard and ProLoan, were running on IMS/ DL1 platforms. The purpose of this project was to

a) shift ProCard and ProLoan over to the new mainframe platform.
b) shift our in-house technical platform over to the new mainframe platform.
c) change the programming language from JSP to COBOL 2.

It was a main goal for the project to come up with an integrated system, which could be packed with functionality according to customer needs. The packages would be the defined products. Architecturally, the products are modularised using building block (called domains) to get a better integration, and to reuse common business functions. This is illustrated in Figure KryTor. 2.

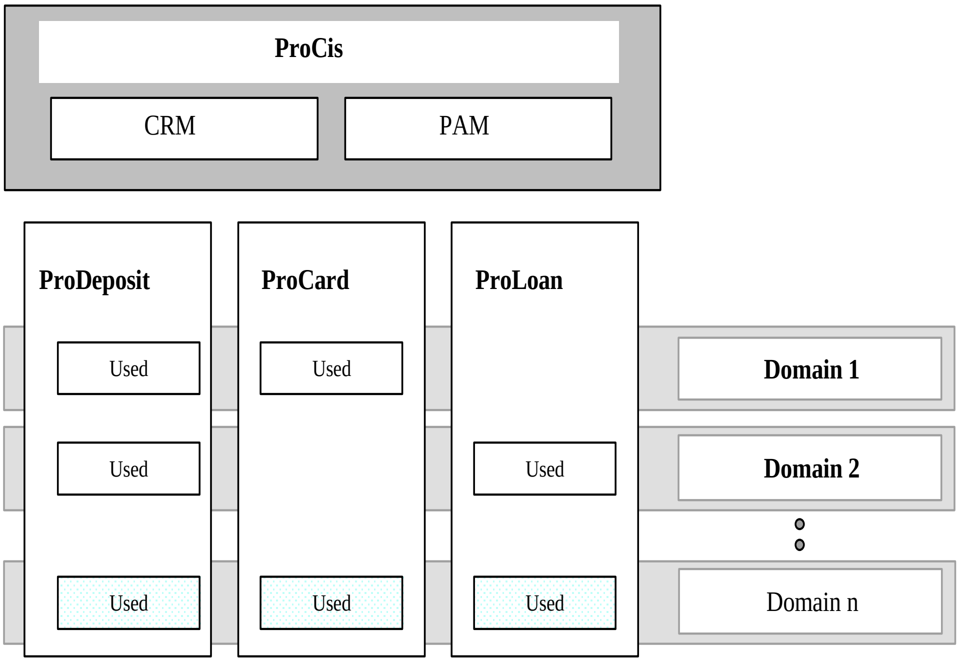


Figure KryTor. 2 Modularization using domains

The baseline project adding Card and Loan to this architecture, had a budget of 3 mill. ECU and involved 25 people. The project was started late May 96 and lasted for about 12 months. Our main focus was on the design phase and a small part of the Cut (Construction and Unit Testing)-phase.

Experiment Overview

The project work of REPRO was planned to be executed in four major phases, running for approximately 3, 6, 6 and 3 months, respectively:

1. Preparation
2. Development for reuse
3. Development with reuse
4. Evaluation, dissemination

Within the phases the work was granulated into work-packages confirming to the key areas of reuse, namely *Organisation and Project management*, *Development* (for and with reuse), *Repository management* and *Metrics and measurements*. For some of the areas, the work was split into several work-packages, in order to have a manageable size. In addition there are specific work-packages for *Project management*, *Evaluation* and *Dissemination*.

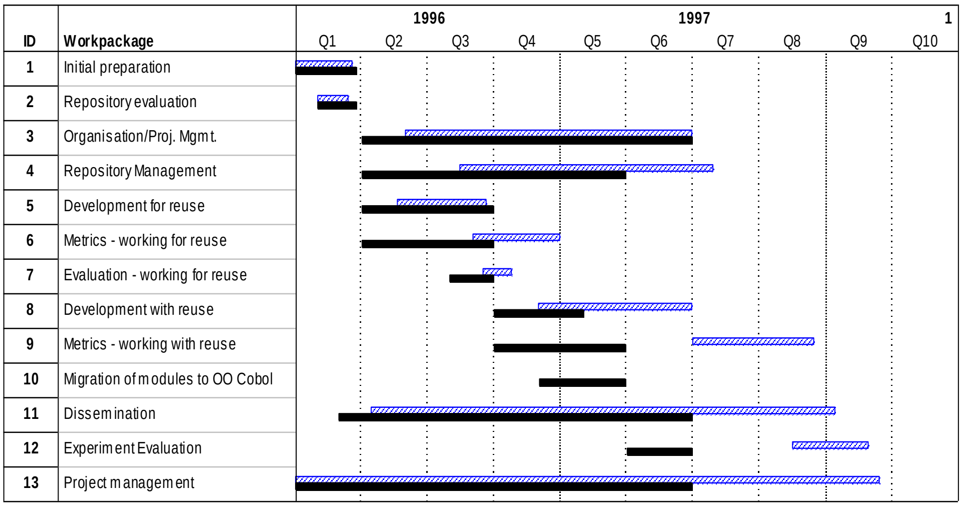


Figure KryTor.3 Project Gantt Chart

In Figure KryTor.3 solid bars are baseline planned work-packages and shaded bars are actual.

Implementation of Improvement Actions

Start of baseline project.

In the first phase of the project *Preparation* we focused on activities defining reuse strategy, roles and procedures. A cost/pricing model was also developed see [ESS, Ch. 13].

Early impact on organisation.

Our main achievement in this phase was the focus REPRO had got in the organisation, with attention from managing director and down to programmers in the product department. The emphasis we had in REPRO on *organisational issues already* made an impact on the organisation as such; REPRO defined new roles that were implemented in our product department - not only in our baseline project. The programming personnel in the product department became *organised* around the domains. This assures high level of competence through all phases of a project. Furthermore, the different groups were capable of supporting several projects at the same time, incorporating requirements from *different* customers.

Repository decision is strategic.

Furthermore, we became aware of the fact that it was not possible to change repository for one single project (i.e. the baseline project of REPRO) leaving the production line behind using the old tools. This is due to the fact that all information in our current repository is inter-related and that introduction of a new tool for a certain area would lead to major migration problems later. Hence, the first conclusion was that to change the tool for repository was a huge job, well beyond the scope of REPRO.

The Repository was planned to contain documentation, metrics, and configuration management information in an integrated manner. At this stage it was decided, for the time being, to continue with the same repository tools as we had used the last years.
For documentation we continued to use DataManager. For Configuration Management we continued to use CCC. To gather metrics – for which we have had no tools earlier - we used a new project repository, PG4.

Two baseline projects merged into one.

To a certain extent we changed the objectives of the second phase. The reuse assessment had clearly showed that a natural way for Provida to achieve reuse is *not* an approach where some projects develop *for* reuse and other develop *with* reuse as outlined in the original plan.
A deeper analysis than the one we could perform the year before showed that Provida's ideal organisation for reuse development was somewhat different than assumed. We typically have *Integrated development for reuse* and are reusing the components in similar projects within the same domain (see section 2.2 of [Kar]). This is also reflected in the organisation which is typically *domain-oriented* (see section 1.4.4 of [Kar]), where a Product department co-ordinates the work of all delivery projects.
Hence, we focused around a *domain* in the baseline project. This analysis secured that the development phases of the baseline project were according to the requirements set by REPRO. Thus we guided the baseline project with respect to:

- Categorising reusable assets from earlier development
- Categorise the ProRetail domain into different sub-domains (this is required in order to market and sell different modules of ProRetail as stand-alone products).
- Defining the procedures for reuse between different sub-domains
- Defining the generic interfaces between these domains

Domain analysis was a new process for our division.

The major adaptation was:
- Variability and generality analysis was made a more comprehensive exercise than originally planned.
- The variability and generality analysis was carried out at the logical level of our process, rather than on physical modules see [ESS, Ch. 15]

Reuse impact on logical level.

At this point savings of 30 % were reported achieved on the logical level. Our reuse strategy was adjusted to address this saving potential focusing stronger on the product owner role. Since our preparations based on REBOOT focused on reuse at the physical level, we had a problem to define proper metrics on the logical level.

Cancelling the migration to Object Oriented COBOL.

It was clear that neither our customers needed it nor was our development department ready for this step. It seemed to be to many unknown parameters for us to cope with at the same time.

New repository for documentation in place at end of the experiment.

Based on our earlier requirements, a new repository tool, Rochade, was bought for the documentation purpose of the repository. It did not, of course, fulfil all our requirements, but was mainly selected due to:

- Availability, running on a PC as a multi-user and multi-session process.
- Low threshold for initial use.
- Easy implementation of the customisation functionality from our old repository.

Since the phase *development for reuse* was shifted more into a preparation phase, the phase *development with reuse* had to be some combination of both populating and harvesting the repository. The repository in use up to this point was the old one running in a main-frame environment. When shifting to the new repository, it was a huge job to customise the new repository to reach the highly customised level we had in the old repository, and training the staff before transporting the content. At the end of the experiment the new repository is in place and gives a much better situation for following projects. However, it was too late to gather experience in this experiment.

Tools

CCC
We continued to use CCC as our configuration management tool. Using this tool gives an accurate track of each physical module, and it enables us to keep track of which version was the actual version at a given time. It also enables us to document changes between versions.

DataManager (DM)
DM has been our product repository manager since the start-up of the company, and we used this tool in a major part of the REPRO project period. Rochade now replaces this tool.

Software Engineer (SE)
SE was used to document the Gaps found in requirement analysis at the customer site. It could be used to create Data Model diagrams and document the requirements in an integrated manner. Moreover, it is integrated with *Systems Engineering* which was the framework for our project process.
The tool was used in our baseline projects in Repro. However, we could not map this description to our system model, which resulted in difficulties later when the documentation should be converted to our repository. Thus it was decided not to go along with this way of working.

Project Gateway 4, project repository (PG4) from Marin Research.
We selected this tool to handle our project administration.
Project Gateway is a system for building and maintaining project repositories using Lotus Notes.

Our intention was to use this tool to gather all the necessary metrics, and also use it as a project information tool. This turned out to be a too ambitious goal.
However, we actively use the tool to publish project schedules made with MS-project. We also use it for the individual project member to report hours used and outstanding on individual tasks. This way we have an almost automatic project schedule tracking, and it has given us detailed measurement of effort used on detailed activities, sometimes down to physical module level.

Rochade
Rochade was selected as our new repository tool based on recommendation from the Repro project. See [ESS Ch. 12]. Converting to a new repository, however, is a huge task. The implementation could not be done within the timeframe of Repro, but the RC division of Provida now has this fully operational.

As a part of evaluation of repositories we bought an already existing report [Ovum]
For details with regards to Rochade we refer to this evaluation.

Measured Results, Impact and Lessons Learned

The Measured Results

Before the development of Card and Loan systems we estimated the reuse potential:
The Reuse potential was calculated using the knowledge on the logical functional level, before a mapping was made available on the physical module level. (In fact, that mapping will not be available until the detailed physical design in the Development with Reuse phase).
On the logical functional level under each domain, all functions that need changes or amendments, and all new functions, were listed together with an estimate of the effort of the change. It was further mapped if the functions were needed in the Card, and/or in the Loan development.
From this mapping, the development cost for both Loan and Card separately and together, was calculated.

The findings from this analysis is shown in Figure KryTor.4:

Estimation for the Cut Stage	Work days
Developing Card Separately	1501
Developing Loan Separately	1060
Sum if separately developed	2561
Development together	1735

Gross Save	826
Variability & Generality Analysis	338
Net Save	488

Figure KryTor.4 : Cost/benefit findings

The table shows that for the cut stage, a net saving of 488 days are saved in developing these systems on the common retail platform, rather than as two separate projects on that platform. This is a net save of 19% if we regard the variability analysis as only needed for this purpose. The fact is, however, that much of the work in the V&G must have been done in each project - totally up to 70%. If we take that into consideration, then 237 days of the V&G was needed anyhow (to describe requirements), and the real save is 725 days, which is 28% of the total.
The real save will be higher because of considerable effects in system test and later on in the maintenance phase.

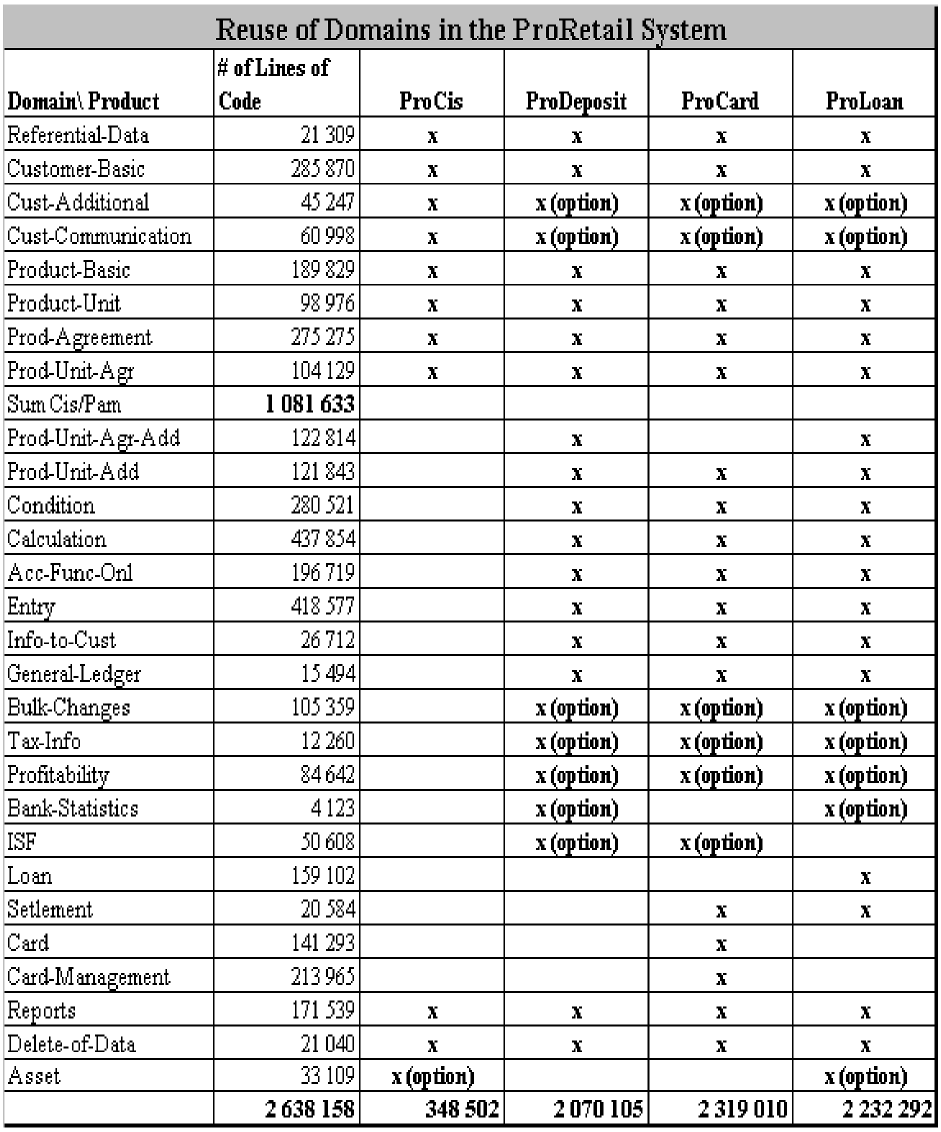
Based on the above analysis, it was recommended (and decided) to develop those systems in common, wherever appropriate.
That is:

- Card was selected since we had a customer contract on that system
- If a task (lowest logical functional description level) was to be changed, re-developed or developed as a new task, we would check the V&G to see if there also were Loan requirements on that task.
- Tasks containing Loan requirement (from the above search) would be developed with the full Card and Loan functionality from the V&G.
- Tasks with only Loan requirement will be left out until we have a customer contract on that product.

Following this procedure will take care of the reuse effect from this development.

After we have developed both the Card and Loan systems, we have run some measures on our resulting (new) repository.

The result is shown in Figure KryTor.5.



Reuse of Domains in the ProRetail System					
Domain\ Product	**# of Lines of Code**	**ProCis**	**ProDeposit**	**ProCard**	**ProLoan**
Referential-Data	21 309	x	x	x	x
Customer-Basic	285 870	x	x	x	x
Cust-Additional	45 247	x	x (option)	x (option)	x (option)
Cust-Communication	60 998	x	x (option)	x (option)	x (option)
Product-Basic	189 829	x	x	x	x
Product-Unit	98 976	x	x	x	x
Prod-Agreement	275 275	x	x	x	x
Prod-Unit-Agr	104 129	x	x	x	x
Sum Cis/Pam	**1 081 633**				
Prod-Unit-Agr-Add	122 814		x		x
Prod-Unit-Add	121 843		x	x	x
Condition	280 521		x	x	x
Calculation	437 854		x	x	x
Acc-Func-Onl	196 719		x	x	x
Entry	418 577		x	x	x
Info-to-Cust	26 712		x	x	x
General-Ledger	15 494		x	x	x
Bulk-Changes	105 359		x (option)	x (option)	x (option)
Tax-Info	12 260		x (option)	x (option)	x (option)
Profitability	84 642		x (option)	x (option)	x (option)
Bank-Statistics	4 123		x (option)		x (option)
ISF	50 608		x (option)	x (option)	
Loan	159 102				x
Setlement	20 584			x	x
Card	141 293			x	
Card-Management	213 965			x	
Reports	171 539	x	x	x	x
Delete-of-Data	21 040	x	x	x	x
Asset	33 109	x (option)			x (option)
	2 638 158	**348 502**	**2 070 105**	**2 319 010**	**2 232 292**

Figure KryTor.5: Reuse Numbers

As shown in Figure KryTor.5, the ProRetail product family consists of ProCis, ProDeposit, ProCard and ProLoan. All of which are now converted to our new environment and technical architecture.

ProCis is used in ProDeposit, ProCard , and ProLoan as well as a stand alone system. ProCis contain 1.081.633 lines of code, and the other modules all together contain 2.638.158 lines of code.
We first developed ProDeposit, then ProCard and ProLoan. In Figure KryTor.6 the percentages of new code are shown.

Product	**Developed new lines of Code**	**Total lines of code in Product**	**% new code in product**
ProDeposit	2070105	2070105	100%
ProCard	375842	2319010	16,2%

ProLoan	192211	2232292	8,6%

Figure KryTor.6: Reuse Numbers

Since the measurement was taken after all systems were developed, we have, however, not counted for the amendment of code in each system. When we developed Card, we amended some domains in Deposit, and when we developed Loan, we amended some code in both Deposit and Card (because of our tactic – as stated above – the amendment to Card was minimal). The real reuse would be lower than the table indicates. However, even if the amendment is as high as 30%, we have a reuse ratio of more than 50% in both the Card and the Loan development.
That is almost twice as high as we expected!

Impact

Organisational Impact

- **Impact on the Organisation**
  A reuse strategy is defined and is continuously improved, according to overall strategies. Furthermore, the strategy is understood and accepted by the management. The organisation is set up to support *integrated development for reuse*.

  If required, internal cost models will be changed to stimulate reuse.

  We also had a goal to establish a reuse board which should on a regular basis review and guide the reuse activity in our company. This reuse board has not been established. Our experience is that reuse must be an ongoing activity in the professional staff, and hence this is one of the obligations assigned to the group leaders for our product domains.

  Also the long-term funding of the reuse activity must be within each project, giving payoff in each project.

- **The Development Process**
  In this paragraph reuse means reusing logical structures, i.e. reusing the syntax of our data model in new domains (or part of domains), or mapping new functionality into existing parts of our model by generalisation of the existing structure. We also reuse general rules and framework from existing functions in new areas.

  Reuse is reflected in all phases of the development process, from requirement specification, through analysis, design and construction to testing. Hence we have:

  - Improved development procedures taking *reuse* issues into account. Furthermore, the procedures will be constantly changed based on results from previous projects. We do not use proper metrics and measurements in these improvements.
  - A more mature organisation, where the roles responsible for reuse (which lies on the domain leader) will see to that the projects deliver results according to long-term

product plans, and not only to satisfy one single customer. This will lead to a more stable core of the ProRetail system. It is anticipated that this core will continue to grow and include functionality from future customers, making a delivery more a matter of configuring the right system.

- **Project Management**
  The reuse experts is represented during requirements specification for all projects, by professional staff for the domain groups concerned. Further reuse is an integral part of the development procedure in our product department.

  The project leaders thus have no special concern for reuse in their projects.

  In our post project reviews we collect reuse information and estimates, but we have not established formal procedures for this.

Cultural Impact

The practical reuse of code and design/documentation in our division before the REPRO experiment, was similar to the situation in many other companies. The experienced designers and programmers were owners of proprietary libraries, from which they selected components – best fited according to their memory of the components - and amended it for their new use, creating a new component. Typically there would be no connection to the original component. In this way of reuse, the most experienced would be the ones “reusing” most, and the novice would have almost no reuse of components.
Maintenance done in one configuration was not built into other configurations because we had no direct reference where we could find out which amendments were needed to those configurations.

The REPRO experiment has changed this behaviour in our division. “Reuse” is now a common known aspect, and is focused on in every discussion regarding productivity. Project planning and start-up activities, focus on how to reuse methods, procedures and components to get the job done in the best possible manner. This reuse thinking emphasises the reuse both on the traditional individual level and on the formal level.

Maintenance is always built into the last common base of the product, and where found necessary – offered to other configurations as well.

Skill Impact

The skill impact is hard to measure. There has been several changes in the organisation in addition to the introduction of new tools and techniques. When too many parameters are changed at the same time it is hard to identify the reasons. However we can state some qualitative scenarios.

- The domain groups have participants in our business requirement analysis.
  This task requires better business functional skills in the development domain, and it has lead to better business understanding giving a more comprehensive design. It is also very valuable when designing test cases.

- The domain groups take part in all stages of our process.
  This gives a skill transfer from our sales and technical personnel to the developers, and also helps disseminate product knowledge from developers to sales personnel.
- The reuse attitude gives broader product knowledge.
  It is necessary to investigate other parts of the product to find reuse candidates. This process helps disseminate product knowledge among the domain groups, and gives a much more flexible staff, being able to work in various parts of the product, wherever the need is.

Because of these changes we now have a better skilled staff. This gives us more flexibility to adjust our development according to changing needs and requirements from our customers.

Key Lessons Learned

As outlined above, we believe there are a large number of organisations that can benefit from the results of REPRO. Especially, organisations similar to Provida can benefit from our results; namely small and medium sized enterprises (SME's) operating in a high-competitive international market. Such companies must work smarter or faster to cope with competition, and we believe *reuse* is one step in this direction.
Companies having most of the characteristics described below, could probably use our results in a Process Improvement Project introducing reuse:

- Have a *defined level* for the system development process according to the Capability Maturity Model, but *low* score according to the Reuse Maturity Model.
- Realise the benefits of reuse, but do not have the means to introduce it to the organisation
- Have the *management's* commitment to introduce organised reuse
- Develop large systems.
- Use Data Modelling as a basic tool for development.
- Have several versions of the same system, but some problems with configuration management.

General

The experiment was planned with a number of updates to our new reuse procedures and guidelines. It was not manageable to update these during the project. We had neither enough experience nor management capacity to do these updates. These deliverables was thus merged with later versions into one update at the end of the experiment. In retrospective view, this is due to unrealistic expectations.

- The change and enhancement of procedures was planned in too narrow time-scales. It was not possible for the organisation to adopt to all the changes, and hence some steps had to be merged.
- It was not practical to start reuse on the physical level before preparing/ structuring the logical level for reuse.

Start the reuse effort on the logical level will mean to reuse parts of "models". Look for similar structures in the data model, and try to repeat this for other parts. Similar business functions were mapped to existing parts of our model, using generalisation of the model to cover for new functionality. Adapting this method requires that the company use data modelling.

To start a reuse experiment on the physical level – like outlined in reference REBOOT – we need an organisation with strict conformance to detailed procedures. It is likely to expect this only in ISO – 9000 certified organisations, or in organisations having a similar quality system. The reason for this is that the change in roles and procedures is a huge step. It is not likely to succeed if all these changes are imposed as one step – the organisation needs time to get used to thinking "reuse". Also gathering all the required metric for each physical module is a huge change, if this is not already part of the development process. The motivation will not be present until the organisation understands and think "reuse"

Technological Point of View

Introduction of repository.

If this is needed for the organisation, it should be anticipated as a major task. It will take time to plan the introduction, to train the staff, and to do the actual conversion. However, an appropriate repository tool is necessary for reuse, either this is on the logical or the physical level

Adaptability.

The adaptability of the organisation to the new paradigm was not as good as we expected. This may be due to the fact that we underestimated the REUSE learning curve and the fact that REUSE is not just another enhancement: it is another way of thinking, working and modelling. It is important to educate managers as well, not only users and developers.

Understanding the REUSE.

The most important skill to obtain, is grasping the concept: what are the important aspects of REUSE, yielding return on the long term investments and avoiding short term optimisations.

TOOL stability.

Introducing new tools require training. It is often underestimated that this will take time. Moreover, it is difficult to introduce many changes at the same time, so introducing new tools should not be done simultaneously with introducing reuse procedures. Necessary tools should be introduced before starting formal reuse procedures.

Business Point of View

Our experience is that investment in proper reuse methods and tools being applied in the right amount and manner is saving us a lot of development and maintenance time. At this stage we have not gathered enough experience to quantify this save, but it is a general accepted statement in the product department.

By referencing to *organised reuse* and a model for maturity we achieve interest from the market.

Once properly learned, we see clear evidence of organised *reuse* giving significantly shorter time to market. Prototypes are more easily built and thus new user requirements met.

New Roles. Be prepared to focus on new roles in your organisation. Roles such as architect, mentor and reuse expert are the most obvious.

Reuse will not happen all by itself. It is essential to incorporate reuse in the standard development process, and thus avoid dropping out after your first project.

Focus on reuse as a long term investment and not as a short term benefit. However, to get top management support, it is necessary with payoff in following projects. Try to achieve integrated reuse.

Secure skilled people. The bottom line is the availability of good skilled people, and they are hard to find. The Norwegian educational system does not typically produce candidates that can be set to work directly. Additional internal education and mentoring is necessary for increasing the skills in a way that can produce a component based-system. We feel that REUSE should be focused in the education at the university.

Strengths and weaknesses of the experiment

Management Commitment.

The top management initially did have a strong commitment and supported the experiment in the early phases. The first deliverable from REPRO, early in the project preparation phase, was our *Reuse Strategy* and *Role and Procedure Description*. These two documents were then implemented in the organisation. This was perhaps the most important single step for the project, and the management felt we had taken a huge step forward to fulfil their intention with the project.

At the start up of *Development for Reuse* we got some trouble in starting up the baseline project, as the contract for the baseline project was not signed as planned in REPRO. This delay also had financial effects of the company, and management attention was shifted more towards the daily operations.

These two events, solving the strategic reuse problem and a difficult market situation, was felt like shifting the focus from the Reuse project. In later phases the project participants felt they no longer had this strong management commitment.

Organisational recognition.

As the problems with keeping up with schedules arose, the recognition of the Reuse project suffered. This may be a general attitude in many companies, and especially affected a project trying to change attitude and behaviour, and introducing new methodology.

Also the effort of the people involved in REPRO was not longer recognised by the company management, and thus needed support vanished.

Process Maturity and Management.

At the end of the experiment we have to admit that our organisation was not mature for this kind of experiment. Despite all the difficulties, we feel that this experiment has contributed a lot to maturity of our organisation, so we are in a fairly good position to succeed in assuring the assimilation of the reuse attitude in our organisation.

We have to take the rest step-wise in a time frame the organisation will accept.

Introducing formal measurements of processes after projects and applying metrics to measure the improvement should be the next steps forward.

In REPRO we introduced strict management tools and methodology to the baseline project. This has resulted in very good tracking of projects in the product department.

Some Traps to Avoid

- Risk Management.
  We did not have a proper Risk Management where the major risk was properly treated.
  - The timeframe between procedure changes was too tight.
  - The timing of the baseline projects did not meet REPRO requirements, and the time frame was too tight to be able to adjust properly.

  We had no contingency plan to compensate for these problems. It is felt that a proper risk assessment in the planning phase could have revealed these risks.

- An organisational point.
  The project co-ordinator must be an operational member of the project team, being given sufficient budgets to participate actively all the time. Being given a budget aimed at only catering for the required co-ordination work is asking for trouble and problems.

  A certain amount of power through a suitable position in the line hierarchy is also important.

- Key competence required.
  We underestimated the learning curve. Partly we focused too much on training in implementation, and accordingly too little on analysis, design and object thinking. As a consequence we had to adapt the approach several times. Partly, mastering a new paradigm sufficiently requires a quite deep understanding of the concepts. Especially so whenever the developer is reasonably proficient in some other paradigm.

Conclusion and Future Actions

The current technical environment

In Figure KryTor.7 our current development environment is shown. PG4, Our test script database and the CR/PTD database all are implemented in Lotus Notes. The experiment have added the PG4 database into this environment to capture effort measures, and the main repository tool is changed.

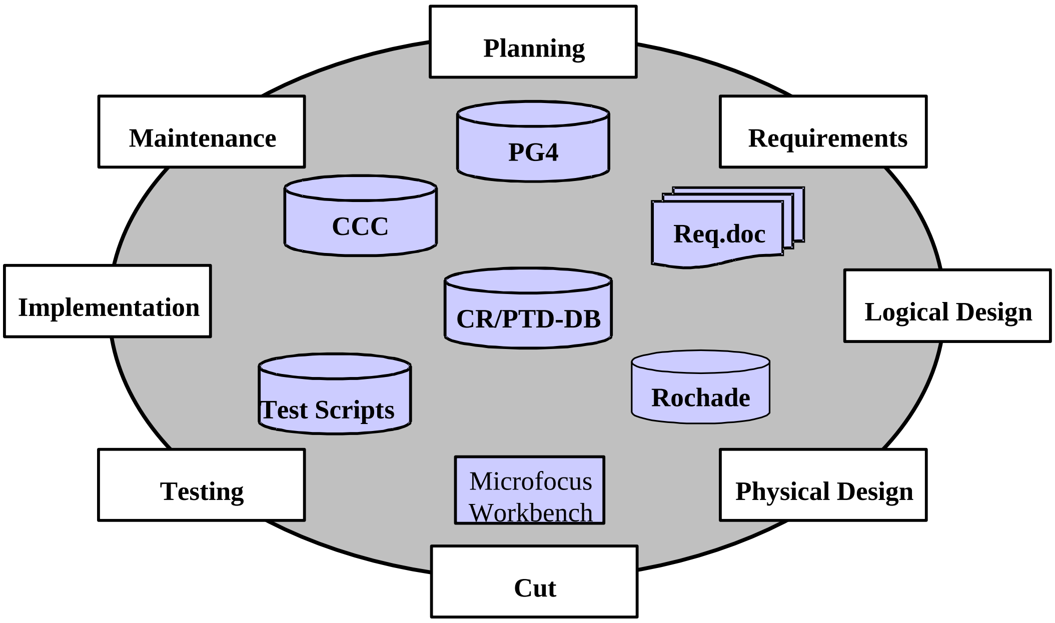


Figure KryTor.7: Tools in our development environment

Organisational Amendments

During this experiment we have improved our organisation with respect to reuse, and introduced reuse roles and responsibilities in the organisation. The target to get to level 3 in RMM is not met on all five key reuse areas. We are on level 3 for both organisation and project management. Also our development process has improved considerably and is between 2 and 3, but in the areas for repository management and metrics we are still about level 2.

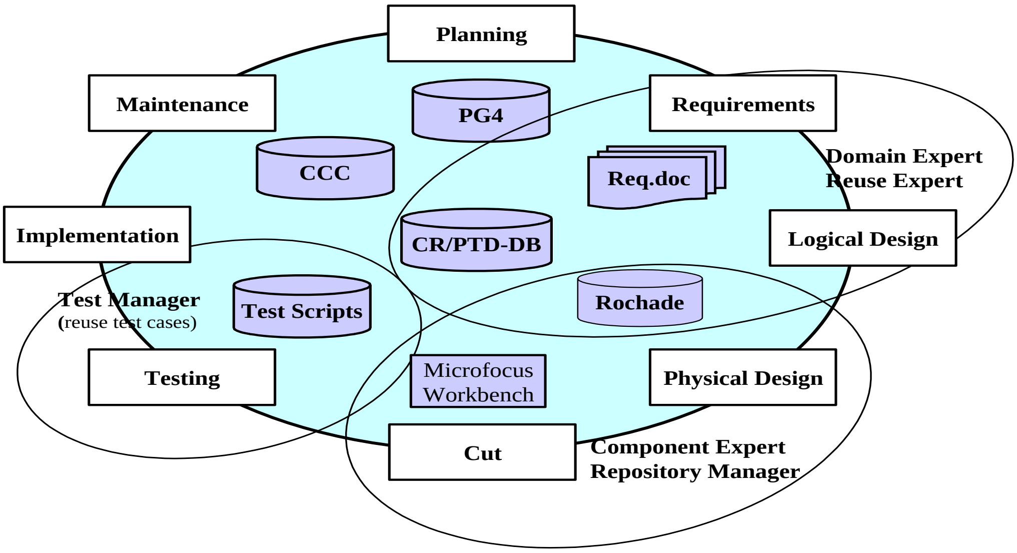


Figure KryTor.8: Main areas of reuse

The main areas of reuse are shown in Figure KryTor.8.

Our way of practising reuse is based upon *integrated development for Reuse,* as described in reference [Kar]. We are currently focusing on the Data Model and our Logical Description, but have also started the process for physical modules. In more details:

- The most important one is the domain expert/reuse expert area, where we have reuse of structures as the main target.
- The component expert/ repository manager area is the traditional reuse where we reuse component. Currently the clean component reuse is in its starting phase.
- On the test side we currently reuse much of our detailed test scripts, and we are working to automate this area..

For the repository we will amend our standard on the logical level, and document more comprehensively the domains and the interfaces between domains.
For metrics on modules we have to find suitable tools to do our metrics, before this is incorporated into our ordinary working practice.
For future amendments we will use a more involving process from people executing the established roles. In this way we expect a more rapid adoption of a new working practice. Also such amendments should be a result of the work done with regards to *reuse strategy* and *role and procedure description*.

Technical Enhancements

Integration of version control system and repository.
Currently an evaluation is going on in order to search for the possibility of moving or version control system from our main frame repository to one on PC. One scenario is to enhance our repository, Rochade, with such functionality.

Build libraries of reusable components was one of our goals in this experiment. This has not been achieved for modules. Currently these modules have a general name, and not a name placing them in one specific domain according to our naming convention.
We will strengthen our architectural role and assign reuse responsibilities to this role. As a starting point we will use generality analyses on components as integrated part of our development procedure. Applying metrics on this level will be introduced on a later stage.

Our development environment is illustrated in Figure KryTor.7. Introducing changes as described above, will bring us a step further to our target solution, illustrated in Figure KryTor.9. However it will still be some future steps to take until we have a totally integrated solution. Figure KryTor.9 is almost the same as anticipated in ref. [Flaa], so we believe there are many case tool suppliers working to reach this solution.

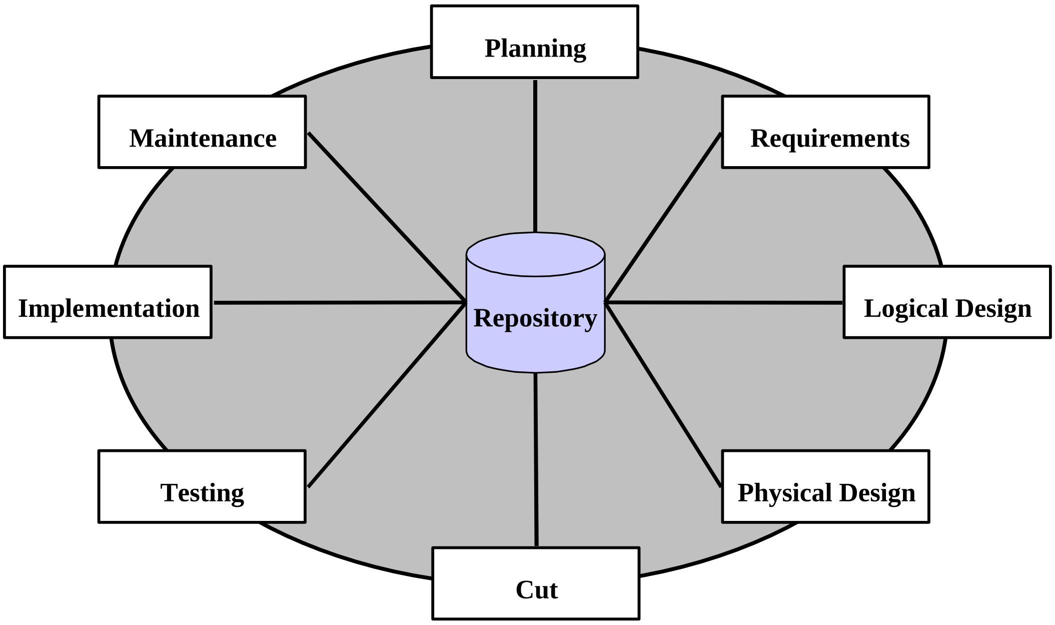


Figure KryTor.9: Target solution for our environment

Business Point of View

The domain concept, dividing our total ProRetail Product into smaller building blocks from which we build or market products, have come out to be a great success. Currently all our products is converted and built this way.

This has both increased our overall productivity and thus reduced time-to-market for our product. As the market products wary in functional scope, this is not easy to estimate. A reduction on about 30% should be a fair guess. We will try to enhance our cost-pricing model and thus “prove” the validity of organised reuse in business.

The improvement in quality will first show over time, as we get experience with maintenance. Currently we do not have any estimate on this area, but we will continue collecting data.

We have decided to take the division into separate domains a step further, making some of the domains especially in our batch part of the product smaller. This will allow for better tailoring the system according to customer need.

Glossary

CMM
Capability maturity model - A general process assessment model developed by Software Engineering Institute. In this paper we use it with extentions defined in chapter 5.4.3 of [Kar]

CM
Change Request – describes a separate change to the system

FCM
Factor-Criteria-Metric – A model for software assessment.
This model is described in chapter 4.3.3 of [Kar]

JSP
Jackson Structured Programming

PTD
Problem Tracking Document – describes a problem (or an error), its resolution and to which configurations the problem is fixed.

RMM
Reuse maturity model – A model to assess the reuse maturity of a company.
This model is described in chapter 5.4.2 of [Kar]

V&G
Variability & Generality Analysis

References

[ESS] Final Report - REPRO, Reuse Process and Organisation improvement experiment, ESSI Project 21513

[Kar] Karlsson, Software Reuse – A Holistic Approach, Wiley, 1995

ISBN 0 471 95489 6

[Ovum] Repositories and Framework, Rosemary Rock-Evans

[Flaa] Foundations of Business Systems, Andersen Consulting Arthur Andersen & Co.

Appendix 1 – Author CV

Mr. Roar Tørlen is a chief consultant at PROVIDA ASA.
He is born in 1947 and graduated from the Technical University of Norway in 1970, Department for Information processing, and has thirty years of experience in developing information systems.
In the years 1970 to 1975 he worked at the Computing Centre research institute at the university with system development methods and data base systems. In 1975 to 1979 he was with Kongsberg Våpenfabrikk, responsible for systems running on minis handling manufacturing logistic. In 1980 he joined Sunnmørsbanken (later merged with Cristiania Bank og Creditkasse) and has since than worked with banking systems.
He has broad experience in many areas as System Analysis, System Design, Data Base Design, Operation Automation, Technical Architecture, Project Management and General Management. He has been a project manager responsible for several system development projects up to hundred man-month of effort.
In 1995 he joint Provida as a project manager in the Retail division.

Mr. Ulf J. Krystad is a chief consultant at PROVIDA ASA.
He is born in 1953 and graduated (M. Sc.) from the Institute of Informatics, University of Oslo in 1982.
From 1982 until 1986 he worked within a Norwegian CAD/CAM company developing systems for the offshore market. From 1986 until 1989 he worked at IDA building financial systems. In 1989 he joined the department of Industrial Mathematics at SI (Centre for Industrial Research, joined with SINTEF in 1995) as a researcher. In 1999 he joined Provida with main responsibility for the development of clients within the Retail division.
His experience is covering different Management responsibilities, a variety of customer related activities including requirements and specification, lectures and presentations especially within mathematical and numerical modelling, system development and process improvement experiments.

Appendix 2 - PROVIDA ASA – a Company Description

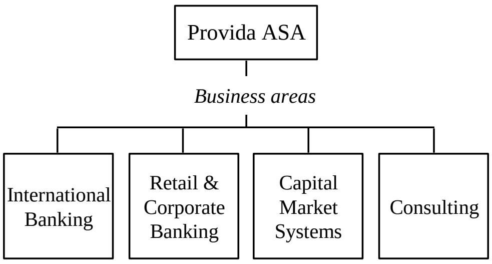
As a leading International Software and Consultancy house, Provida have served the needs of the Financial and Banking Industry for over 30 years. Through creative systems, harnessing modern technology, we have established ourselves as a market leader for these products.
Provida has a wealth of technical and business knowledge, gained through long associations with major financial institutions. With in-depth knowledge throughout the organisation and with the strength of our systems, Provida strives to give our customers the competitive edge needed to succeed in the banking and financial industry.

Provida- Business Areas

Provida develops and sells software solutions and consultancy services to the International Banking and Financial Market. The vision of the company is that Provida will, via its system solutions, contribute to the overall profitability and efficiency of the bank and financial institution.

The main business areas and therefore product offerings are best summarised by referral to the following diagram:

Provida ASA – Business Areas



Retail & Corporate Banking Product Suite

ProRetail represents a new generation of banking systems, and has been developed with a market-oriented banking philosophy and modern system architecture. Emphasis has been placed on the design to provide a structure which, as well as addressing today's business requirements, allows a cost-effective integration of essential new business functions and technologies in the future.

The solution includes up-to-date banking applications for the personal and corporate market. The core systems in ProRetail consist of a central customer, agreement and product administration system plus systems for administering loans, deposits and cards.

Priority is given to real-time access to all customer information. This enables the financial institution to have continually updated information concerning agreements between the customer and the financial institution, the customer’s financial status and all other aspects of a customer’s relationship with the financial institution.

By using the Product Warehouse, the financial institution can create and develop new banking products on-line. This provides a unique opportunity to promote the right products to the market at the right time. This solution enables the financial institution to launch aggressive sales strategies, identify groups of customers with specific needs, and carry out banking operations at a far lower cost than was previously possible.

SEPIOR - Practical Experiences with Reusable CAM Components

P. Bininda, A. Blessing, W. Daxwanger,
T. Krenzke, O. Schmid
SEKAS GmbH
Perchtinger Straße 3
81379 München, Germany
http://www.sekas.de

K. Bergner, A. Rausch, M. Sihling
Institut für Informatik
Technische Universität München
80290 München, Germany
http://www4.informatik.tu-muenchen.de

Introduction

Recently, the componentware development paradigm has gained much attention. On the one hand, approaches like COM or JavaBeans promise to boost the performance of application developers, creating a fast-growing market for start-up companies especially in the areas of GUI design and desktop computing. On the other hand, vendors of large enterprise systems like SAP R/3 are planning to implement modular versions of formerly monolithic software systems.
In this experience report, we provide an example for the commercial, technical, and human implications of componentware in a different context, namely, a department of the company SEKAS [1] specialised in projects for computer-aided manufacturing (CAM) systems. The main purpose of CAM systems is to control the fabrication process from raw materials to final

products. This task requires the co-ordination of a variety of different activities: 1) the calculation of production schedules, 2) the control of production lines, machines, and transport facilities, 3) the management of resources and tools, 4) the gathering and logging of machine data and 5) the management and propagation of errors and alarms. Modern, highly automated CAM systems can do all this with no or only minimal interaction by human users.

In this report, we describe the considerations and experiences of SEKAS during the Process Improvement Experiment (PIE) SEPIOR [2], which is partly sponsored by the European Systems & Software Initiative ESSI [3]. SEPIOR is an acronym for "Software Engineering Process Improvement through Systematic Application of Object-Oriented Techniques and Reusability". Consequently, the specific goals of the SEPIOR experiment are:

1. Enabling the reuse of pre-fabricated software components by systematically introducing object-oriented technology. This will reduce development time and costs for customer-specific solutions.
2. Increasing the quality of the customer-specific solutions through the use of reliable components as building blocks for individual solutions.

After outlining the initial scenario in the remainder of this section, the expected strategic and commercial aspects of componentware are described with the focus on the involved chances and risks. The following section then sketches parts of the development process for the adoption and introduction of the new techniques. Based on this, the subsequent section covers a practical application of the described process exemplified by the development of an alarm management component. Finally, some of the lessons learned during the process are given, concentrating mainly on the human impacts of the adoption.

Company Context

SEKAS is a small software company founded in 1988. It currently employs 40 employees, of which over 85% are specialists in software engineering and computer science. Activities are mainly focused on the European market, and sometimes also address the world-wide market. SEKAS offers high-end products and quality services to middle-sized and large manufacturers mainly in electrical engineering and electronic business. About 75% of the turnover is gained with customer-specific development of sophisticated technical and scientific applications. The remaining turnover is achieved with high-end products for quality management and automated testing. The current success of SEKAS relies on the following cornerstones:

- Highly skilled and trained employees with much practical experience.
- Focus on market segments where the key competences of SEKAS can be optimally applied.
- State-of-the-art development environments.

Motivation for the PIE

Despite its current success, SEKAS does not yet fully exploit its potential for productivity and economic success. Although most of the software engineers at SEKAS are trained in object-oriented techniques, the current level of software reuse is rather low. This is mainly accredited to the fact that no systematic introduction of an overall OO-software engineering process has been performed yet. This would include process modelling as well as organisation, training and coaching.

The need for improvement in this area arises from the growing competition caused by the emerging software engineering countries (India, Romania, Russia, etc.) For the future, it is mandatory for SEKAS to cope with this challenge by providing increased value for the money spent by the customers. This implies further improvements with respect to development process as well as software quality.

Strategic and Commercial Aspects

The development of CAM software is a very demanding task, as it requires knowledge in distributed system architectures, and the ability to implement fail-safe software for a variety of different real-time controllers and devices on heterogeneous platforms (cf. [4], [5]). During the last decade, SEKAS has gained profound insight and large experience from a variety of CAM projects. The acquired knowledge in the areas of embedded systems, real-time software, and production control systems, as well as the achieved standards of software engineering and quality management are regarded as the key competences of the company.

Despite this successful history, experience has also indicated some recurring problems, including the lack of standardised technical infrastructures. Therefore, even software realising basic functionality had to be developed from scratch multiple times. This pertains, for example, to the implementation of several low-level communication libraries. The upcoming of standardised infrastructures, protocols, and components will render such efforts unnecessary, allowing SEKAS to accelerate system development and to concentrate on its core business. On the one hand, this shortens the time-to-market for the customers of SEKAS. On the other hand, it also leaves more time for SEKAS to realise additional features. Furthermore, the use of standard infrastructures is expected to have a positive impact on software quality and interoperability with other systems.

Another critical issue is that the reuse level within SEKAS is currently rather low. Essentially, every production control system was built from scratch, tailored to the actual customer requirements and technical infrastructure. Although the ability to adapt to different technical infrastructures is seen as a strength of SEKAS, there is also consensus that a higher reuse level could considerably raise the productivity. If SEKAS succeeds in developing reusable, high-quality components independent from specific technical infrastructures, the effort for developing different versions of the same functionality for different infrastructures will be significantly lower. This will reduce the development costs and raise the competitiveness of SEKAS in customised software projects. To achieve this goal, a clear separation between domain-oriented modelling and technical modelling has to be achieved. Consequently, the domain model represents those parts of the components that are independent from a specific technical infrastructure. This way, the domain model can be reused or even directly mapped to several infrastructures (cf. [6]).

In the long term, this strategy enables SEKAS to gradually transform into a component vendor, selling products on the emerging CAM component market. The shift from custom development projects to products naturally requires careful preparation, as it necessitates a variety of additional capabilities and organisational measures, for example those regarding marketing and customer support.

The main risk in the scope of the sketched componentware strategy is the uncertainty about possible pay-backs for the costly development of reusable components. To reduce this risk, a careful analysis and selection of suitable components is necessary.

Furthermore, most components evolve from actual projects which usually do not care much for reusability due to the general lack of development resources and time. Thus, the decision to build a generic component has to be backed up by adequate funding, resources, and organisational means.

To further evaluate the described approach, SEKAS has decided to set up the PIE SEPIOR. A team consisting of in-house domain experts and experts in OO development was formed. This team was supported by experts in object-oriented methodologies and componentware techniques from Technische Universität München, who acted as consultants and coaches. In a first step within SEPIOR, existing parts of a CAM system were refined and modelled as reusable, platform-independent components.

Process Model

As a solid foundation for the development of components a suitable process model has to be introduced. In this section, we present an architecture-centric, iterative, incremental, and reuse-driven software development process that has been successfully integrated within the SEKAS methodology. The main goal was the elaboration of a component-based architecture that is flexible enough to be adapted to and reused in numerous applications of a common domain. A software architecture of this kind basically consists of two parts: 1) a set of components and 2) an underlying common framework gluing those components together. The framework provides standardised interfaces, classes and communication mechanisms, and thus allows the components to interact with each other in the scope of a predefined structure. Especially the components within this framework serve as a point of adaptation and configuration. Components can be conveniently adjusted or even replaced without touching other components within the framework.

The presented process consists of four essential activities:

1. Identify components and describe their functionality.
2. Specify business-oriented component requirements, model component interfaces, and design the interactions between them.
3. Design the underlying common framework.
4. Design the technical architecture and select a corresponding infrastructure.

Usually, these activities are interleaved and performed in an iterative and incremental fashion. Note the clear separation of business-oriented aspects and technical activities (second and fourth activity, respectively) in this process. This is an essential requirement as stated in Section “Strategic and Commercial Aspects”.

Identify and Describe Components

Basically, there are two fundamental approaches for identifying the components of a system: On the one hand, there is the more traditional “top-down” approach in which the overall system is decomposed into components according to the specification of the corresponding requirements. On the other hand, emerging component markets suggest the reuse of prefabricated components. Therefore, a “bottom-up” approach aligns the system’s design to the specification of appropriate, existing commercial or in-house components as much as possible. Usually, in practice both approaches are combined in the context of an iterative and incremental overall process. Such a process leads to an understanding which components are to be developed from scratch and which ones can be reused (from previous projects or as bought on a component market).

Ideally, the set of components to be developed corresponds to the key competences of the company. Each of these components should be further investigated as a candidate for further reuse in other projects or even on a commercial component market. This task requires a great amount of experience and domain knowledge to decide whether the additional effort in creating a reusable component will result in an appropriate pay-back later on (for instance, as a result of ongoing reuse). If so, even more effort needs to be put into widening the component's functionality. A good starting point are the experiences gained in previous projects. To get a comprehensive overview, a company might consider further, unfulfilled demands and requirements from clients, having a look at competing products, if they exist.

The result of this activity is the "big picture" of the domain under consideration. It includes a set of components to be developed, a set of components to be bought and integrated, and a description of their functionality and interaction.

Specify Component Requirements, Model Interfaces and Interactions

After the components to be developed have been identified, a detailed analysis of each component's requirements and its relations to other components is carried out. This involves modelling the interfaces of all involved components using common description techniques like UML [7] as well as performing walk-throughs for selected use cases. Another main goal of this step is to balance the level of abstraction of the component. If the abstraction level is too high, a lot of work is needed for adaptation; if it is too low, the component is not likely to be reused in other projects. There is no common solution for this problem. It depends on the experience of the developer to find the right level of abstraction.

The results of this activity incorporate for each component a set of the use cases it is involved in, as well as specifications of its behaviour and its interfaces. For this, graphical description techniques as offered by the UML can be used.

Design the Framework

The activities described above result in a set of abstract, business-oriented components. For most applications, this is not sufficient. They also need some common facilities that cannot be assigned explicitly to a single business-oriented component. This pertains, for example, to foundation classes that are used by a number of co-operating components, or to base mechanisms like persistence management. Furthermore, the framework may encompass certain guidelines that have to be observed by all components or interfaces.

The main result of this activity is a class diagram specifying the framework and implementation classes for the components under development.

Design Technical Architecture

The last main activity is to capture the requirements for the actual technical architecture, and to select a corresponding technical infrastructure consisting of suitable middleware components. Usually, the separation of business-oriented and technical design leads to a clear architecture, as technical details of a certain infrastructure don't influence the business-oriented parts. Furthermore, this approach allows the reuse of a certain business-oriented design for multiple technical infrastructures.

The main result of this activity is a specification of the technical components of the system, including their interfaces and the interaction between them. Furthermore, the mapping from the business-oriented components and concepts to technical components and mechanisms must be provided.

Application Example: Alarm Management System

This section demonstrates the application of the process model described in the previous section at the example of an alarm management system (AMS) component. This component is specified and implemented at SEKAS in order to serve as a reusable building block in current and future projects.

Identify and Describe Components

Following the process model, a total of seven CAM projects were analysed as a starting point in several workshops and design sessions at SEKAS. As a result of the analysis process, some candidate components were identified whose functionality was needed in multiple projects. Together, they represent a large portion of the company's core competences. Figure SEKAS.1 shows a distilled and highly abstracted version of the identified components.

In our model, a superordinated ProductionPlanningAndControlSystem (PPCS) component deals with the commercial aspects of the respective fabrication process, for example, product pricing and customer orders. Product information for the necessary planning, scheduling, and optimisation of production tasks is available from the ProductManagement component (PM). The obtained information constitutes a part of the individual production plan for a certain product. Furthermore, a production plan includes data like a bill of materials, production steps, machine set-ups, and so on. The PPCS delegates the computed set of optimised tasks to the LineControlSystem component (LCS) in form of orders for production lines and machines.

The LCS initiates and subsequently controls the processing of a production line order. Depending on the degree of automation, the LCS may control the machines with or without using the ResourceManagementSystem component (RMS). All products manufactured by the LCS are managed and tracked by the PM throughout their whole lifecycle. The LCS usually not only collects the product tracking data, but also gathers machine and production data.

Finally, the AlarmManagementSystem (AMS) component serves as a kind of global service and may thus be used by any component. The AMS is used to inform users about system errors or problems occurring during the production process. The AMS is a suitable component for demonstrating the approach introduced in the previous section due to its importance; its functionality was necessary in five of the seven projects analysed. Furthermore, to the knowledge of the authors there is currently no commercially available AMS component that fulfils all the requirements comprised in the subsequent section.

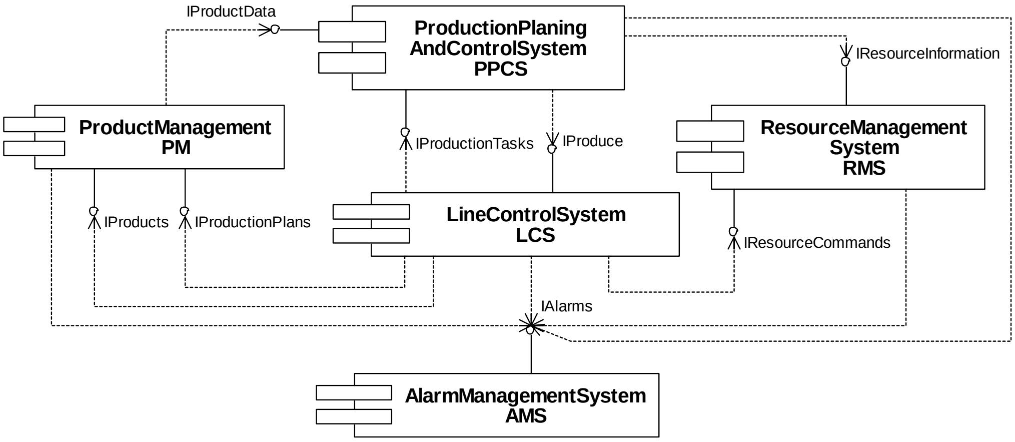


Fig. SEKAS.1: High-Level Component Architecture of a CAM System

Specify Component Requirements, Model Interfaces and Interactions

A CAM system is by its nature a distributed system. There are components running on a wide range of different heterogeneous platforms distributed throughout a factory. The platforms include, for example, embedded real-time systems on production machines, real-time systems for production control, database systems, and Windows and UNIX workstations.

Every component in the CAM system is a potential client of an alarm management system, since every component can encounter events that need attention. However, it is not desirable to have alarm handling done locally on every machine. An AMS should be manageable as a single instance and should marshal access to central resources such as pagers and phone lines.

In order to emerge an appropriate design for a reusable AMS component, it is necessary to define the requirements for the architectural design, the framework and the implementation. The basic requirements are:

1. *No loss of error information*
2. *Asynchronous error handling*
3. *Freely configurable error handling with escalation strategies*
4. *Platform independence*
5. *Transparent API*
6. *Context-sensitive error classification*

Figure SEKAS.2 shows the major parts of the architecture of the AMS including the framework to connect external systems to the AMS.

In detail, the External System 1 and 2 represent components that use the AMS for informing its users about errors, problems or events appearing during the runtime of the external system (for more details see Section “Design the Framework”).

The AMS in figure SEKAS.2 consists of two instances of the AMS-component, namely AMS 1 and AMS 2. Each of them accepts errors from external systems and decides which strategy is applied to handle the error according to its configuration. A strategy specifies which error is handled by which device handler and states the parameters that are given to the handler.

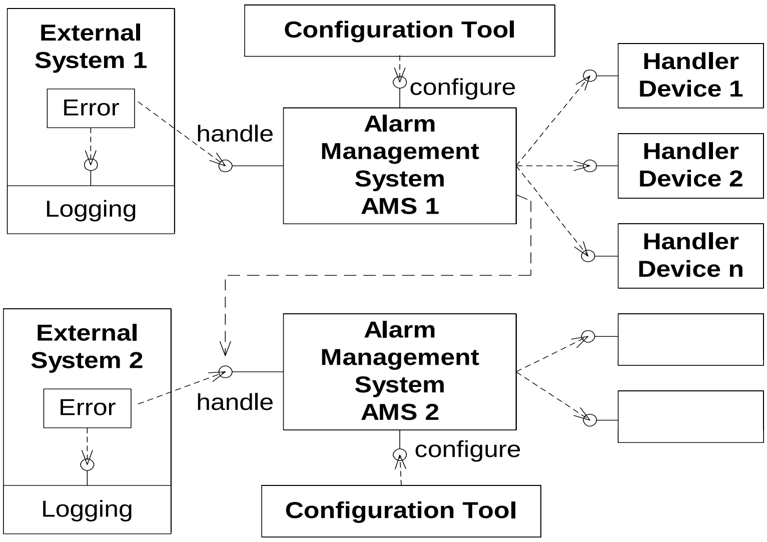


Fig. SEKAS.2 : Architecture of the AMS.

According to requirement 2, the error handling is done asynchronously, so that the external system is not blocked and does not suffer from any overhead imposed by the error management (for more details see Section "Design Technical Architecture").

The core AMS gets the information about success or failure of the error handling in the device handler. According to requirement 3, an escalation of this error is necessary in case of failure of the error handling. Escalation means the automatic re-handling of the error with another strategy.

Strategies are defined by the user of the AMS and are imported into the AMS by the configuration tool. The decision to separate the configuration from the AMS has been reached due to the fact that it should be possible to re-configure the AMS during runtime without interfering the actual error handling. Because of requirement 1, no loss of error information is allowed. Therefore, a minimal error handling independent from the configuration is provided. This error handling has to be configured regarding the installed system.

The device handlers are those parts of the AMS that actually inform the user of the error. A device handler can be a pager that reports critical errors to the user, or a printer that logs important but not critical events. We decided to locate the device handler outside the core of the AMS component so the user can add additional device handlers just by starting them and re-configuring the AMS.

Design the Framework

To use the AMS component in an efficient and easy way, it is necessary to design a practicable framework for the clients of the AMS as a part of the overall framework gluing together the components of a CAM system. Since the external system cannot connect to the AMS without the framework, the framework has to be designed very carefully.

According to requirement 5, the framework must provide a transparent API. It is needed for the external system to use the AMS in an easy and therefore efficient way. Simultaneously, any misuse of the AMS through the external system must be prohibited. The shallow interface "handle", shown in figure SEKAS.2, fulfils this requirement. Essentially, this interface is sufficient for sending errors. No further knowledge about the configuration and the possible error handling is necessary.

The AMS fits into a system wide error handling mechanism, so that it is not necessary to include different error handling mechanism in the external system. Therefore the basic AMS error class is located in the framework, and the external system can derive its own error classes from this basic error class. The external class can now use the error for internal error handling and furthermore pass the error to the AMS. According to requirement 6, the basic error class allows a context-sensitive classification of an error according to its severity. Furthermore, a re-classification in a different context is possible.

The framework provides a functionality to log error information for example for debugging purposes. This mechanism prevents that the AMS is blocked with debugging output. The design allows the use of the framework without a connected AMS. This way a simple error handling and logging mechanism is provided.

Design Technical Architecture

As stated above, various components throughout a factory must be able to access the AMS. To enable communication, an appropriate infrastructure has to be established. The traditional approach within most CAM systems is to use TCP/IP sockets with a proprietary protocol in order to handle the communication between the components. This approach implies high development efforts and does not encourage reuse. The alternative is to use a standard infrastructure technology. We have chosen CORBA [8], because it is platform and language independent and widely available. Consequently, the AMS component offers its functionality via CORBA interfaces to its clients. The communication of the AMS component with its device handlers also employs CORBA.

CORBA was chosen over DCOM [9] because it is an open standard with implementations available for all relevant platforms, while DCOM is primarily available on the Windows platform. In addition, production control and industrial applications are a traditional domain of CORBA, while DCOM focuses primarily on desktop components.

As stated previously, the CORBA interface of the AMS is language independent. Therefore, clients can communicate with the AMS regardless of the language in which it is implemented. However, as error handling is an integral part of the implementation of any component, language specific extensions of the framework are necessary. These framework extensions are available for Java and C++, which allow clients to handle errors and to trace information with minimal programming effort.

The anticipated performance bottleneck of the AMS is the network communication between the client component and the AMS on the one hand, and between AMS and the alarm devices on the other hand. Therefore, the efficiency of the programming language used to implement the AMS is not an major issue. Consequently, we have chosen to implement the AMS in Java although we expect inferior performance compared to other programming languages such as C++. We encountered a significantly shorter development time than in C++, especially since memory management and the integration with CORBA are much less complicated. Based on experiences from a preliminary project[1], the use of Java saved an estimated 30% in development time. Thanks to the choice of Java and CORBA, the platform neutrality stated in requirement 4 is also implicitly fulfilled.

Lessons Learned

[1] The project comprised components written in both Java and C++ with various CORBA implementations.

As mentioned above, the first step towards component development was the domain analysis of the CAM domain. Although the participating staff at SEKAS already had basic training in OO analysis and design, it was not easy to adapt this rather abstract knowledge to the concrete request to do a domain analysis. The training has to be complemented by suitable guidelines for the development process. Otherwise, the developers will be unsure where to start the analysis and whether the analysis covers all critical sections later on. Coaching from OO and componentware experts should be employed until the practical use of the learned techniques is adopted by the team members.

For the domain analysis it is necessary to have experts for the problem domain and experts for OO techniques. According to our experience, they do not need to be the same people.

We also learned that the ideal size of an analysis team is three to five members. One or two members eventually forget critical details, more than five members sometimes linger in endless discussions, drifting away to technical details instead of concentrating on the domain problems. We also discovered that it is necessary to cut such discussions from time to time, and to restart them with a smaller team. These results have an organisational impact on the planning of the person power for future design activities.

During the whole design and implementation phase of the components a CASE-Tool with UML-support and sophisticated code generation was used. We think it is not practical to make the design without such a tool. It is also necessary to use the tool during implementation, because an iterative development cycle is only possible if the way from design to implementation and back to design is feasible. That implies that the tool supports good synchronisation mechanisms between the source code and the design model and good code generation possibilities. In fact, code generation and synchronisation really shortens the implementation time and makes the design more robust, because the design is not hidden in source code and so the developer is prevented from destroying good design during an implementation enthusiasm phase.

We also found out, that a CASE-Tool using UML helps very much in providing documentation and keeping it up to date. The reason for this is, that the UML diagrams are easy to understand and are kept up to date, using the synchronisation techniques of our tool.

During design and development we experienced that the more time we spent to design the important parts, the less time was needed for implementation and the more readable and simple was the emerging source code.

However, it is not necessary to first design every detail of the component and then proceed to implementation. On the contrary, it is sometimes necessary to make a detailed design of the important parts and a very rough design of the less important parts, to implement the system prototypically, and to try whether the design works. Then one can go back to design and specify the missing parts. If it is found that there are conceptual errors in the design, it will cost less time to make corrections according to this approach.

Result Measurement

The assessment of the degree of reusability is done based on a so-called base line project. The base line project comprises selected parts from an actual project of a representative SEKAS customer from the CAM environment. The customer project included the integration of distinct manufacturing lines, transport, logistics and test into an continuous production process. Customer acceptance was reached in May 1999.

The effect of reuse is assessed by comparing the effort needed for the original development of the base line project (delivered to the customer) and the effort needed for the redefinition of the base line project using the newly constructed components and their framework. The goal is to reduce development effort by at least 20%. An analogous comparison is drawn regarding warranty and maintenance efforts. These should be reduced from 5% of the original development costs to 2.5 %.

The introduced measurement avoids the need for prototypical projects, but takes some time to get a statistical relevant result for maintenance efforts.

The expected commercial impact cannot be consolidated at the moment, but will be assessed at the end of the project. However, the authors are confident in the success of SEPIOR, because the lessons learned so far are more than positive.

Conclusions

Although the process improvement experiment SEPIOR is not completed the time this paper is written, some preliminary conclusions can be drawn. The conclusions mainly summarise the technical and human aspects in introducing component based software development.

Despite the initial expense of introducing component based development, the reuse aspect permanently spreads at SEKAS, indicating the rising acceptance of these techniques and methodologies. One of the major impediments is to create continuous stimuli to foster the development of reusable components on the one hand, and to emphasise the deployment of the components on the other hand. The primary risk of management activities inciting these stimuli is that developers overshoot. A natural balance between developing reusable components and specifically customised modules or prototypes has to be found. This premises a skilled developer, who not only shows technical excellence and experience but also possesses common sense. Thus, in the authors opinion the management activities should emphasise on continuous professional technical and social training of the developing staff instead of financial or non-financial incentives.

The major technical issue is the development of a framework for components both during component development and component deployment. This topic is most important for component vendors. The component framework has to include aspects such as communication, configuration, persistency and distribution. Additionally, a framework has to comprise testing and debugging aspects that even work with no introspection possibilities based on the code of the components.

References

[1] Homepage of SEKAS GmbH, http://www.sekas.de/, 1999.

[2] Homepage of the SEPIOR project, http://www.sekas.de/, 1999.

[3] Homepage of the European Systems and Software Initiative, http://www.cordis.lu/esprit/src/stessi.htm, 1999.

[4] Smart Fabrication Verbund CIM, Fraunhofer Gesellschaft, IPA, http://smartfab.ipa.fhg.de/, 1997 (in German).

[5] Manufacturing DTF, OMG Document RFP mfg/98-07-05 v2.6, 1998.

[6] Klaus Bergner, Andreas Rausch and Marc Sihling, *Using UML for Modeling a Distributed Java Application*, Technische Universität München, technischer Bericht TUM-I9735, 1997.

[7] Unified Modeling Language Specification, Version 1.3, Object Management Group, http://www.omg.org/, 1999.

[8] CORBA overview, Object Management Group, http://www.omg.org/corba/, 1999.

[9] DCOM overview, http//www.microsoft.com/com/tech/DCOM.asp, 1999.

Authors Information:

Dipl.-Inform. (FH) Paul Bininda studied Computer Sciences at the Fachhochschule München from 1985 to 1991. His graduation project was a portable Oberon compiler and an object oriented GUI-Framework implemented in Oberon. Since 1992 Mr. Bininda is employed at SEKAS GmbH. There, he is the System Manager responsible for the companies heterogeneous technical infrastructure and has been working on a wide range of projects including: Development of the device layer of a radio monitoring system, development of a Motif GUI for a radar surveillance system, test management for emulators, simulators and architecture models of a new super computer, project management for the development of an automatic test case generator/verifier, development of parsers and transformers for different programming languages, development of a scheduler and visualisation process on INMOS Transputers, development of a high performance test data archive and evaluation software for aeroplane turbine manufacturers.

Dipl.-Inform. Andrea Blessing studied Computer Sciences at the Technische Universität München from 1989 to 1995. Since 1995 Mrs. Blessing is employed at SEKAS GmbH. She has been working in projects of different domains such as the design and implementation of a graphical user interface for production test systems, development of a snmp proxy for managing concentrator access multiplexer equipment in a backbone network, development of a configuration tool based on a oracle database for developing graphical user interfaces for measuring devices, development of a distributed security system for online maintenance in a manufacturing company. She is currently second project manager of the SEPIOR project and consultant for object oriented analysis and design at SEKAS.

Dr.-Ing. Wolfgang Daxwanger studied Electrical Engineering at the Technische Universität München from 1986 to 1992. From May 1992 until December 1998 he worked as a research assistant with the Laboratory of Automatic Control Engineering at Technische Universität München. During this time he was concerned with the development of locomotion platforms and computing systems for autonomous mobile robots in the Sonderforschungsbereich 331 „Information processing in autonomous mobile robots.“ For his dissertation on automatic visual parking control using artificial neural and fuzzy networks he received the Dr.-Ing. summa cum laude in July 1999. Since April 1999 Dr. Daxwanger is employed at SEKAS GmbH. His current work is concerned with the design of reusable software components. His major interests are object oriented software development, componentware and softcomputing.

Dipl. Inform. Thomas Krenzke studied Computer Sciences at the Technische Universität München from 1989 to 1995. Since 1991 he worked as a developer for SEKAS GmbH in several Projects. After his graduation in 1995 he was employed at SEKAS GmbH and was involved in the design and implementation of large database systems within the CAQ/CAM domain. He was project manager for the development of a large production control system for the production of printed circuit boards. Due to his experience in the CAQ/CAM area, he is one of the domain – experts in the SEPIOR project at SEKAS GmbH. So he‘s currently working on the design and implementation of reusable components for the CAQ/CAM domain. His major interests are componentware, object-oriented software development and the design and development of database systems.

Dipl.-Inform. Oliver Schmid studied Computer Sciences at the Technische Universität München from 1989 to 1995. From July 1995 to December 1997 he worked as research assistant with the Laboratory of Real-time Systems and Robotics at the Technische Universität München. Between 1992 and 1997 he was involved in the Sonderforschungsbereich 331 „Information processing in autonomous mobile robots.“ There, he was one of the responsible developers for a distributed, object-oriented, active knowledge base with real time capabilities. Further, Mr. Schmid did fundamental research on the topic „Assembly sequence planning with co-operating manipulators.“ Since 1998 Mr. Schmid is employed as software engineer at SEKAS GmbH. His work concerns the area of automation (CAQ/CAM). Under this scope he participated at the development of a production control system for the production of printed circuit boards. Currently he is working on the development of a reusable alarm management component and its final valuation within the SEPIOR project.

Dr. Klaus Bergner studied Computer Sciences at Technische Universität München from 1986 to 1992. Since February 1992, he works as research assistant at the Chair for Software & Systems Engineering of Prof. Dr. Manfred Broy. His dissertation about the development of graphical modelling techniques for object-oriented systems was finished in 1996. Since 1997, Dr. Bergner leads the project FORSOFT A1, which is concerned with component-oriented software development.
Dr. Bergner managed various academic software projects, among them the development of a web-based database system and a distributed CASE-tool. His main interest areas are object-oriented and componentware development methods and software architecture. Since April 1999, he is CEO of the new-founded technology start-up company 4Soft, specialising in componentware development and large business architectures.

Dipl.-Inform. Andreas Rausch studied Computer Sciences at Technische Universität München from 1991 to 1996. Since February 1997, he works as research assistant in the project FORSOFT A1 at the Chair for Software & Systems Engineering. His field of research comprises software architecture, distributed and component-oriented systems, object-oriented modelling and development, and methodological aspects of software engineering.
Before his job at FORSOFT, Mr. Rausch managed various industrial software projects for distributed information systems. He is one of the co-founders and owners of the start-up company 4Soft.

Dipl.-Inform. Marc Sihling studied Computer Sciences at Technische Universität München from 1991 to 1996. He is currently working as research assistant in the project FORSOFT A1 about component-oriented software development. His main interest areas are object-oriented and component-oriented software engineering and graphical description techniques. Mr. Sihling is one of the co-founders and owners of the start-up company 4Soft.

Session 4
SPI and Requirements Management

Chairman

Richard Messnarz

ISCN, Dublin, Ireland, Graz, Austria

The user requirements elicitation and specification process: deployment experience of CSELT DOMINO 2.0 methodology

Antonella Bartocci, Marina Melchioni
CSELT, Via Reiss Romoli 274
10148 Torino, Italy
antonella.bartocci@cselt.it

Introduction

In the area of user requirements management CSELT has developed a methodology called CSELT DOMINO (Distributed Object-oriented Methodology for and

INcremental apprOach) which is now at its version 2.0 ([2], [3]). We started elaborating and experimenting it since 1996 and it has now been applied to over 30 service oriented applications for user requirements elicitation and specification. Our company was new to such an approach so a supporting process has been built to help deploying this methodology. To pursue specific quality objectives our company adopted the ISO9001 model, obtaining this certification two years ago: the cultural change we need to perform is very broad and the feeling is that we've only just started with it. This paper describes CSELT experience on this issue by providing an overview of CSELT DOMINO 2.0, then by describing how it was decided to support its deployment in software projects and the lessons learned from this experience.

About CSELT ..

CSELT, founded in 1964, is the Telecom Italia Group's Company for study, research, experimentation and qualification in the field of telecommunications and information technology. The Center has the technical know-how for the principal activities of the Telecom Italia Group and applies them in the research for new services, advanced applications and integrated solutions, working mainly according to the view of an Operating Company. CSELT contributes to the study of advanced systemistic scenarios, plays a link role between academic and applied research, thanks to its strong presence in the international context, becoming reality through the participation in common research programs and standardization activities. CSELT's lab constitute a reference point for both feasibility and integration studies, development and experimentation of advanced solutions, evaluation and qualification of products and processes, demonstration of innovative services. Furthermore, great attention is paid on in-field systems experimentation, especially in those areas where innovation can be applied in a short-medium time scale.

.. and the Telecommunication Industries

Telecommunication industries are evolving rapidly and changes are often unpredictable: liberalisation and competition put stronger constraints on the time-to-market requirements for faster service delivery. At the same time, these new services are requested to be customisable on one side, and to realise easy and secure access to shared information, on the other side, increasing the use of network resources. All such "environmental" changes require software systems to become always more flexible and integrated with one another. Consequently, complexity of telecommunication applications is always growing: the requested functionalities are increasing in number and they must often be provided by etherogeneous systems. To build an application, especially if it is distributed, it really becomes necessary to be supported by 'a good set of rules' useful to formalize the process of user requirements capture (elicitation and engineering) and for the corresponding development. This is in order to gain control over developing 'the right system', according to the customer needs (it's important to get all requirements and to get them right as early as possible in the software life cycle): in other words, the availability of a methodology useful in gaining control over the complexity of a certain application problem becomes necessary, allowing to analyse it according to different levels of detail, according to different points of view, according to the customer feedback, so that the entire software project management becomes more systematic.

CSELT was originally a center for the medium-long term research while lately the market trend has become such that it has become more focused and tightly drived by its customer specific needs, on tightest time schedules. Typically today our projects are such that we often act as software suppliers towards our customer without actually being completely a software house ourselves, nor do we have such a mandate. In fact, very often we realize our customer software applications either by outsourcing some part (or even all) of the software development or by buying some kind of specific software development consultancy from some external software house. This is the basic reason why historically we have never needed a corporate approach for managing a software project and why we've been gradually changing direction over the last years.

CSELT's Approach to User Requirements Specification over the last years

Due mainly to historical reasons, CSELT has never had a uniform approach to capturing and documenting user requirements, since this was never felt as a specific need before. In fact, CSELT has never had a specific 'organizational culture' towards this activity (nor towards software engineering in general) since our 'products' have always been results of research studies, and even when software was delivered it was accepted for it to be produced without having to follow any specific software life cycle, with no explicit activities defined.

The new trend, previously described in the Introduction for telecommunication industries, is quite challenging for CSELT and it has determined for us quite a radical change since our major software projects now have become more focused and finalized to delivery on a short-to-medium time scale, also changing our 'traditional' way of managing relations with our customers. We are often being commissioned to deliver software applications dealing through its entire software life cycle process, that is, starting from understanding user requirements and going through to software delivery, maintenance and evolution, and (in some cases) also through providing some operational user support.

Our customer's organizational and competitive environment typically change very rapidly and they may impact on some (or all) of the needs previously defined for a certain software system. Also, our customer seldom is able to define specific requirements precisely since he is obviously mainly aware of the major, very high level needs that need to be solved urgently by the software system. While this in the past was not much of a problem, since there was 'plenty' of time to understand explicitly all user requirements, to study and experiment different alternatives and different technical solutions, nowadays this is no longer affordable.

Another somewhat critical issue that we are facing lately is that often we acquire software development consultancy from external software houses, and it happens that some consultants leave before the project has come to its end, meaning that we have to manage such turn-overs, and the difficulties of reducing the cost of coping with them may be enhanced by defining some 'good organizational practices' that all software projects should follow.

As a matter of fact, all these 'environmental' changes are quite challenging for CSELT and to pursue specific quality objectives our company adopted the ISO9001 model, obtaining this certification two years ago. Furthermore, to better understand and focus on processes and activities to improve, and for the reasons just mentioned, concerning the area of software project management, a specific project team has been set up whose tasks can be mapped on the KPAs (Key Process Areas) of CMM level 2 [1]; among these internal processes there is also user requirements management (other issues considered by this team are concerned with: configuration management, testing, software project management, external software suppliers rating and management).

CSELT DOMINO 2.0: what is it and how it was developed

CSELT DOMINO 2.0 is a methodology originally developed to cover, in CSELT's software projects, the user requirements specification and the software requirements specification phases ([4], [5]).

CSELT's need was to define a corporate process to provide a common way for people in a project team to perform the same actions in the same order, for making the same kind of choices, for producing the same documents in certain times, and so on. Also, the need was for a process that is a systematic way for operating in a repeatable manner (i.e., stable) and a common definition of the documents to be produced (contents + structure). The need was also to increase control over the software to be developed starting from the specification phase and to use "a tool" for representing information in a rather rigorous way, reducing ambiguity, redundancy and incomplete coverage of user requirements.

Furthermore CSELT's objectives were to develop a methodology supporting requirements elicitation and engineering in order to produce verifiable specifications, with least ambiguities, increasing completeness and augmenting characteristics such as consistency, modifiability and traceability. Among the objectives there was also the need to support requirements analysis for service oriented systems, in order to define an application as a set of components, to clearly define the application interfaces and to define a first draft of the application architecture. This kind of approach should have augmented software and specifications reuse.

With this set of requirements, the approach adopted was to go for an object-oriented based methodology, looking at the de facto standard techniques of that time (mostly Jacobson [6], Rumbaugh [7] and Booch [8]) and to merge and customize these approaches according to CSELT specific needs. As a matter of fact, these OO methodologies were not found completely adequate for our purposes, since for our software projects we had the specific requisite to provide support starting from the very initial activities of the analysis phase, while those OO methodologies provided good modelling techniques mostly for performing detailed analysis. For historical reasons, as mentioned above, not every department in CSELT is familiar with the main acitivities of a software life cycle, so people are generally not used to specific tasks such as capturing and documenting user requirements, making explicit all implicit requirements, then tracing them through the testing phase, and so on. Moreover, people are generally not aware about all the possible different types of user requirements they should capture, so our urgent need was for a methodology providing guidelines for assuring an exhaustive coverage of user requirements starting from considering the application from the external users point of view. In [6] the Use Case technique was found useful but not enough powerful to provide such support so we introduced a model also inspired by domain analysis for performing the first steps of the user requirements capture activity, which is CSELT DOMINO 2.0 main focus (starting from the user requirements specification developed applying our methodology it is possible to perform detailed analysis and design using a specific object-oriented methodology, or even a different kind of methodology).

Additional general characteristics of DOMINO 2.0 are the fact that it supports an iterative and incremental life cycle and that its architectural framework refers to OSCA [9] and TINA [10] approaches, meaning that it provides guidelines supporting design separation of user interface specific aspects from application logic aspects and from data management aspects (in OSCA these were called user interface-layer, process-layer and data-layer): making design decisions keeping them separated provides advantages for the later activities in the software life-cycle (e.g., maintanance).

Briefly, DOMINO 2.0 is structured as follows (see Figure 1): it is organized in two main phases, User Requirements Specification and Software Requirements Specification. Each phase is then organized in a set of tasks, and for each task we have defined its activities, its inputs and outputs. For each activity DOMINO 2.0 provides a set of guidelines and modelling techniques to perform the task and to produce its associated output information. For the entire process DOMINO 2.0 provides a pre-defined set of templates, customizable, to structure information in the User Requirements Specification and to uniquely identify requirements.

Since CSELT experience is such that software is mostly outsourced (especially for medium-big sized products) we have so far been able to apply mainly the first phase of DOMINO 2.0 so this paper is focused on it.

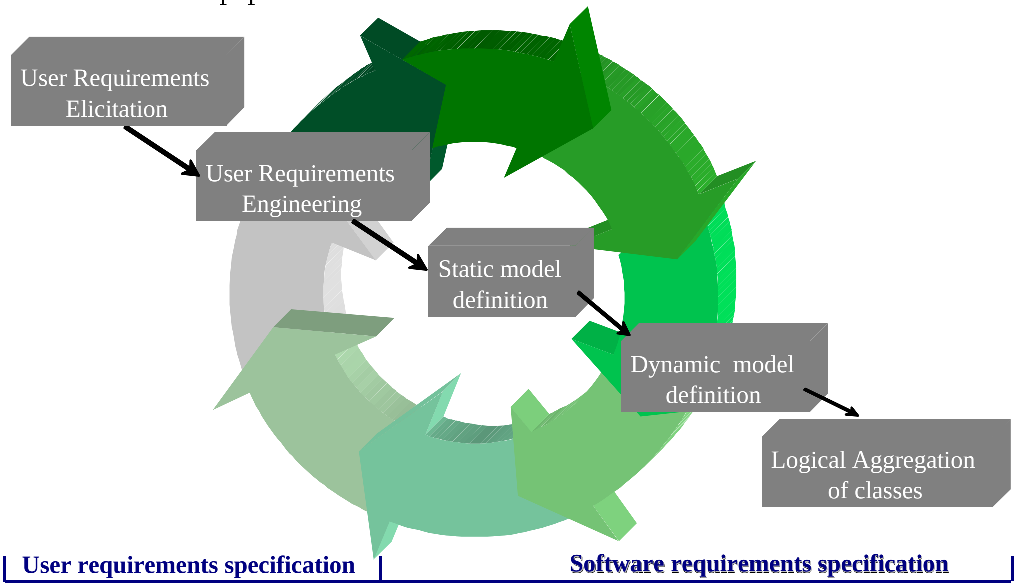

User Requirements Elicitation
User Requirements Engineering
Static model definition
Dynamic model definition
Logical Aggregation of classes
User requirements specification
Software requirements specification

Fig. BARTOCCI.1: CSELT DOMINO 2.0 for User and Software Requirements Specification tasks.

In DOMINO 2.0 the user requirements specification phase is structured in two tasks:

- User Requirements Elicitation: this task is made of the following activities:

  Application Definition; Definition of Scenarios; Application Context Analysis; Structuring the application services into Components (by performing the Application functional structure analysis and the Identification of components for each service activities); Application increment definition. Note that these activities are generally performed according to an iterative process (rather than sequential).

The goals of this task are mainly to provide a definition of what the application should and should not do, of the relationships/interactions taking place between the application and the entities identified in its external environment (or 'context'), of its main services and a sketch of the application (and service) components.

The modelling techniques used in this task were partly inspired by domain analysis and, in DOMINO 2.0, they are called: *context model* (one diagram for the entire application: it defines the application boundaries, the external entities and the interactions taking place among them. This new modelling technique was developed and introduced in DOMINO since the Use Case model resulted to be less rich of semantic information useful at this point to depict the application characteristic in one diagram and from the external users point of view, *functional structure model* (one diagram for the entire application: it defines the application basic processing components, through a functional decomposition structure; the leaves of the graph in DOMINO 2.0 are called 'process macro-elements'), and *macro-elements model* (one diagram for each application service: it combines the external and the internal functional views, i.e., the two preceding models, providing the initial architectural definition for the application. This model defines which process macro-elements are used to realize the service considered and it also defines which kind of interface macro-elements are needed. In DOMINO 2.0 there are three types of interface macro-elemnts, one for each possible type of external entity: human user, database and network element). DOMINO 2.0 also strongly supports use of *scenarios* to identify and characterize relationships between the application and the external entities in its environment and the application services.

This task produces as main output the application scope definition, an overview of its identified services and the identification of the application increments and releases.

- User Requirements Engineering: this task is made of the following activities:

  Application user requirements specification; Services user requirements specification; Macro-Elements user requirements specification; User Requirements Specification document Finalization. Note that these activities are generally performed according to an iterative process (rather than sequential).

  The goals of this task are mainly to capture and formalize: the application behavior for each application service, the application non functional requirements and the services and macro-elements functional and non-functional requirements.

  This task usually drives back to the previous one in a refinement process, so the modelling techniques are the same. The requirements formalization activities performed in this task are supported by introducing a template with specific fields (customizable to some extent) and numbering rules providing unique identification of each single user requirement. This formalization scheme was introduced to support features such as requirements classification of importance, verifiability, modifiability and traceability. This task produces as output the functional and non-functional user requirements definition (at the application level, services level, process and interface macro-elements level), and the user requirements specification document ready for sign-off (i.e., customer validation).

The Process Implemented for Supporting CSELT DOMINO's deployment

After experimenting the methodology on a case study software project (based on a real software project realization) CSELT DOMINO 2.0 has been gradually spread in our company starting with just a few number of selected software projects, looking for 'friendly' users (i.e., colleagues quite familiar with software engineering concepts and aware of the fact that this approach was not yet 'istitutional' in the organization). The kind of support provided in this phase for learning and applying the methodology was that of providing at the beginning a general overview of the entire methodology approach and then by teaching it in more detail while the project was on-going, giving new elements when they were needed. This experience has been useful both for refining the methodology description itself and also for tuning the deployment approach.

The approach adopted to train future users of this methodology was to provide courses to colleagues on a request basis. The course provided has now come to its eight edition and refining it has been an "on-going" work. At the moment it is structured in three days and it provides an initial general overview of the importance of quality management in a software project, covering its main aspects, then relating these issues to the user requirements specification process and then going in the details of the DOMINO 2.0 methodology.

In parallel with this training activity a number of colleagues have been specifically trained to provide support directly to software projects with a mentoring activity (this support has also been provided on a request basis). The mentoring activities involved have been modelled in a process (see Figure 2) in order to clearly define and separate the mentoring activity (white color) with the project specific activity (grey color). The "tools" made available for supporting the corresponding activity are shown on the right side of the process. The 'Checklist' has been developed to support the mentor in verifying, at certain points in the mentoring process, the correct 'formal' interpretation and application of the methodology concepts and constructs (covering all its aspects, from the models to the chapters to be included in the final specification document); all other consistency doubts/checks need to be worked out together with someone from the project team. The 'Tool' refers to a customization that was applied to some market tools (RequisitePro and SoDA) to support requirements management and the automatic generation of the user requirements specification template according to DOMINO 2.0.

Since DOMINO 2.0 is based on an iterative process, the mentoring process also results to be iterative. The mentoring process activities shown in Figure 2 are:

- ❑ *Explaining the problem*: at the beginning, the person being supported has to briefly explain what the application to be developed is about. This generally happens through a few meetings and some documents exchange.
- ❑ *Understanding the problem*: the mentor needs to gain a sufficient understanding of the application characteristics in order to be able to provide adequate support in applying DOMINO 2.0 to that specific project.
- ❑ *Customizing DOMINO 2.0 application for the specific problem*: here the objective is to determine which will be the user requirements specification characteristics according to how this document will be used. Considerations in this

activity are mainly concerned on issues such as fixing the level of detail that the analysis should have, what fields of the requirements template need to be used, and so on.

- *User Requirements Elicitation*: this task is supported by mentoring but, as shown in Figure 2, the activities being specifically supported are those related to building the models (while defining and writing user requirements in the document are activities left up to the project team).

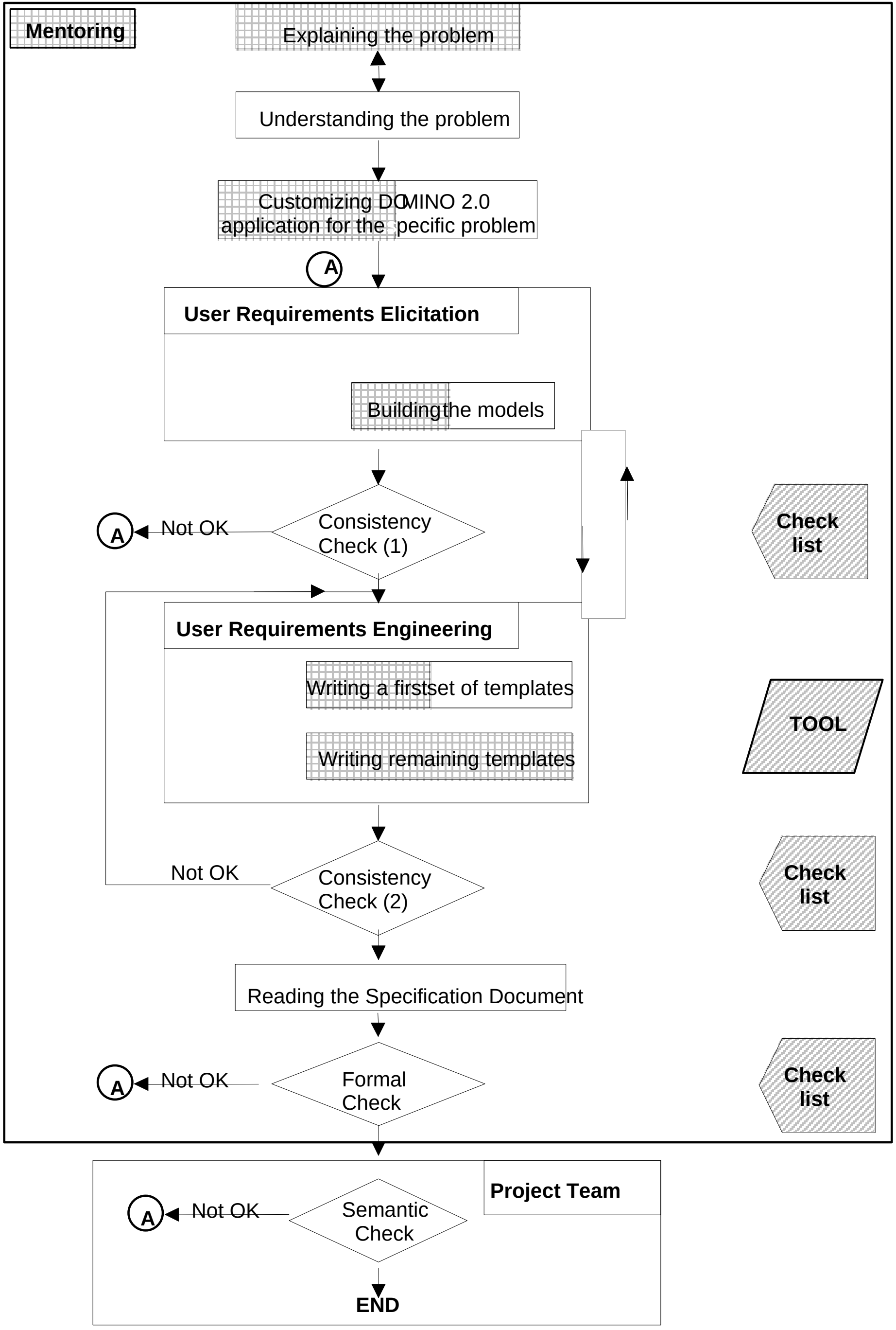

Mentoring
Explaining the problem
Understanding the problem
Customizing DOMINO 2.0 application for the specific problem
A
User Requirements Elicitation
Building the models
Consistency Check (1)
Not OK
A
Check list
User Requirements Engineering
Writing a first set of templates
Writing remaining templates
TOOL
Not OK
Consistency Check (2)
Check list
Reading the Specification Document
Formal Check
Not OK
A
Check list
Project Team
Semantic Check
Not OK
A
END

Fig. BARTOCCI.2: the mentoring process for CSELT DOMINO 2.0 deployment.

- ❑ *Consistency Check (1)*: this check goes through the models developed in the User Requirements Elicitation task in order to verify that they've been correctly understood and applied by the project team; this check also verifies 'formal' consistency among the different models. The mentor uses the Checklist here and typically performs these checks together with someone from the project team. This kind of consistency check is generally performed more than once and it is performed the first time when the models start to be stable.
- ❑ *User Requirements Engineering*: this task is supported by mentoring but, as shown in Figure 2, the activities being specifically supported are those related to formalizing user requirements by filling the templates provided by DOMINO 2.0. Here the project team may choose to use the Tool to be supported in managing user requirements templates and in generating the Word document.
- ❑ *Consistency Check (2)*: this check goes through the templates filled in the User Requirements Engineering task to verify if they have been correctly applied and also checks 'formal' consistency between these and the models previously developed. The mentor uses the Checklist to perform this activity.
- ❑ *Formal Check*: this is to verify that formally the final user requirements specification document is structured according to DOMINO 2.0 and according to the customizations that were defined at the beginning. This check is performed by the mentor.
- ❑ *Semantic Check*: this is to validate that the final user requirements specification is in line with the customer needs. Of course this kind of check can only be performed by someone from the project team (typically the project manager) and it is actually an on-going activity, throughout the specification process. Here it is highlighted just once, in a specific point, as a quality check before asking the customer a final and formal validation of the document and for then allowing exiting this phase of the project.

Lessons Learned ..

As already mentioned, due to historical reasons CSELT has never needed a corporate approach for managing a software project but lately we've been gradually changing direction. DOMINO 2.0 is the first methodology for capturing and specifying user requirements being spread in our company (it is not the only one, though) and this is only one aspect of the whole matter. The cultural transition we are performing is broader and in fact there's a specific project team providing support to software project managers by selecting/developing and introducing in CSELT a set of guidelines. All these guidelines are now being integrated in the definition of a basic set of activities that any software project life cycle in CSELT should have. Another aspect of the integration being performed, for example, relating to DOMINO 2.0, is that guidelines have also been developed for defining test cases for user requirements specifications generated by applying this methodology.

.. about CSELT DOMINO 2.0 and its deployment approach ..

DOMINO 2.0 application overall is giving a positive feedback. The major advantages appreciated by users are that it offers a structured and guided approach to gradually approach the complexity of the specific problem, and the context model results to be a powerful and sufficiently userfriendly technique to discuss the application boundaries and services not only within the project team but with the customer also. Team working and cooperation were also found to be improved by the adoption of this methodology thanks to a better organization and documentation of requirements and by using "a common language", also allowing for easier walkthroughs and reviews with the customer for sharing the application vision. Finally, another useful characteristic of DOMINO 2.0 was found to be that it supports producing a specification document with different levels of details so that it is quite easy to extract from it a specification document containing the 'right' level of detail necessary to be validated by the customer while giving the entire specification document to the development and testing groups. Another advantage found by our users was that this kind of component-based specification approach augments specification reuse (new releases may be specified easily with least redundancy simply by referring to previous versions of the user requirements specification). Regarding the User Requirements Elicitation task, it was generally felt that the activities in this task are a good "tool" for enforcing exchanges of ideas between the different roles involved in the project. Regarding the User Requirements Engineering task, it was found useful especially for formalizing user requirements when software development was outsourced and for deriving some test cases. The major disadvantage DOMINO users talk about relates to the overhead of documenting user requirements (which typically may change very often and very rapidly on-going) and in such situations DOMINO 2.0 ends up in requiring too many formalisms and overhead in updating coherently the requirements specification.

As far as the course is concerned, it was initially structured in order to provide colleagues with a case study extracted from an application of DOMINO 2.0 to a real software project. The target in doing so was to comment directly with the class how DOMINO 2.0 specifications were going to look like, how the models are to be applied and so on. This approach though resulted to be less effective than expected since it was hard to make people understand that customizations are possible and also where and when they can take place. The major refinement to the course objectives was then in this direction: it was decided not to show any specific example but rather to work directly in class on a case study, performing in small groups the task's activities. This resulted to be much more effective, providing room and suggestion for broader discussion and questions on particular concepts.

About the mentoring process it was found positive in itself, though there were some drawbacks/refinements necessary. First of all, when people new to the methodology are supported, it becomes harder to trace the line between supporting understanding of the methodology and actually performing the analysis of the specific application, this sometimes resulted in going beyond the planned schedule for the time assigned to the mentor. The major cost of supporting a project in using DOMINO 2.0 was generally felt to be in checking and walking through the analysis models and specification being produced: the mentor is generally not skilled in the specific project domain and yet she/he needs to have some knowledge of it to understand if the modelling techniques are being applied correctly. Also, when the methodology is being used for the first time, the initial models are sometimes completely built together with the mentor. So we looked for some sort of trade-off solution, and what we're thinking is to train the mentoring people in order to cover the different department areas, trying to shorten the gap between the methodology and the domain knowledge in the person supporting deployment. Another relevant finding was that specific time needs to be planned and scheduled ahead by the project manager to take into consideration all the above issues. Furthermore, once a person has become familiar with DOMINO 2.0, by having applied it to at least one project, then subsequent experiences need very little support. For this reason two kinds of supports are being issued lately: a “first level support” (mentoring projects that will apply the methodology for the first time) and “second level support” (mentoring projects already skilled on the methodology but still requiring some ‘spot’ kind of support on very specific issues).

As far as the Checklist is concerned, while it was originally thought to be a tool supporting the mentor in performing the consistency and formal checks of a correct DOMINO 2.0 application, they actually became tools supporting the analysts in producing the specification, especially the models (as if it were a sort of “quick reference guide”), so it is actually being used in both these ways. One difficulty with it is that it becomes hard to apply when the methodology is being customized too much.

DOMINO 2.0 is resulting adequate for medium/large software projects (referring to complexity, cost, number of releases per year, time schedule, control, ..), while resulting in too much overhead for medium/small software projects. For these latter kinds of projects we developed (late last year) a more general set of Methodology Guidelines supporting the user requirements specification process, containing general principals, a small and simple set of suggested modelling techniques, and a specification document template.

.. and about the cultural change that CSELT is performing

Over the past year there have been may occasions to realize that the cultural change that CSELT needs to perform is really complex and for this reason it is being slow. In CSELT this year about 50% of the total number of projects are software projects; the DOMINO 2.0 courses so far have been attended by about 25% of the people allocated on a software project and so far we have supported with mentoring 4 projects with a ‘first-level support’ and 3 with a ‘second-level support’; also other 7 projects asked support in using the more general Methodology Guidelines mentioned above. These are considered quite good numbers to start with, but the reality has been that only in those departments where the top-management was convinced and sponsored a more systematic approach to software engineering, we were really able to have some concrete impact, otherwise we found most people reluctant to perform this change. Another good result and impact for us has been that our System Quality department has modified some of its procedures adopting several results that have been produced by the project specifically set up to support software project management. Among them is the must that every software project from now on will have to document user requirements by producing a specification for them, either by applying DOMINO 2.0 or the more general Methodology Guidelines, and that they will have to provide Test Plans correspondingly. On the other hand, due to historical reasons, CSELT was never institutionally used to thinking and working ‘by processes’ but this new trend is now gradually being adopted by our System Quality department, and the area of software project management will be among the first to be involved by it. For the above reasons in CSELT we don’t have yet a specific metric to measure our improvements (we only keep track of the number of non conformities we get by each of the ISO9001 certification inspections), though another very ‘empirical measure’ we are having, relating to software project management, is that many progress reports of the projects being supported with CSELT methodologies and corresponding mentoring mention that they have produced the user requirements specification using such methodology, or such mentoring (.. people is starting to talk about them).

References

[1] ISO9001: *Quality systems - Models for Quality assurance in design/development, production, installation and servicing*.

[2] A. Bartocci, C. Gajetti, P. M. Maccario *DOMINO (Distributed Object oriented Methodology for an Incremental apprOach) versione 2 - La fase di Specifica dei requisiti utente,* Documento Tecnico CSELT dicembre 1998.

[3] A. Bartocci, C. Gajetti, P. M. Maccario *Template del documento di specifica dei requisiti utente (versione 2) sviluppato secondo la metodologia DOMINO 2.0,* Documento Tecnico CSELT dicembre 1998.

[4] A. Bartocci, P. M. Maccario *Proposta di approccio metodologico per l'analisi di applicazioni software orientate ad oggetti - La Metodologia DOMINO*, Documento Tecnico CSELT dicembre 1996.

[5] Bartocci A., Grasso E., Maccario P.M., Martini G. *Proposta di approccio metodologico ed architettura di riferimento per i sistemi di supervisione e controllo*, Documento Tecnico CSELT, 1996.

[6] I. Jacobson *Object Oriented Software Engineering - A use case driven approach*, Addison Wesley 1992.

[7] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen *Object-Oriented Modelling and Design*, Prentice Hall 1991.

[8] Booch, G. *Object Solutions Managing the Object-Oriented Project*, Addison-Wesley 1996, ISBN 0-8053-0594-7.

[9] Bellcore; *The Bellcore OSCA™ Architetture*, Bellcore Technical Advisory, 1992.

[10] TINA-C deliverable *Computational Modelling Concepts*, 1996.

About the Author

Antonella Bartocci received her degree in Computer Science in 1993 from Turin Universivity. Right afterwards she joined CSELT where she was involved in the Information Technology and Software Architecture Solutions department. She has been active in the ITU-T SGX for standardization of formal specification languages (mainly SDL and MSC) and she worked actively in the integration of the GDMO formalism with SDL. In 1996 she joined CSELT's Software Design and Product Quality Support departement where she became involved on working with object-oriented software analysis and design methodologies, and corresponding CASE tools, to provide support inside CSELT to specific software projects in the different telecommunications applications area. Over the last two years, she was actively involved in the definition, training and mentoring of CSELT DOMINO 2.0 methodology and of the more general Mehthodology Guidelines produced in CSELT for user requirements capture and specification. This recent working experience has also naturally brought her to work with CSELT’s System Quality department, in relation with CSELT’s ISO9001 certification, for defining and refining some of the guidelines and procedures developed specifically for software project managers, in order to start considering them from a software process improvement point of view.

About CSELT

CSELT, founded in 1964, is the Telecom Italia Group's Company for research, experimentation and qualification in the field of telecommunications and information technology. The Center has the technical know-how for the main activities of the Group and applies them in the research for new services, advanced applications and integrated solutions, working mainly according the view of an Operating Companies. CSELT contributes to the study of advanced systemistic scenarios, plays a link role between academic and applied research, also by being present in the international context. CSELT's lab are a reference point for both feasibility and integration studies, development and experimentation of advanced solutions, evaluation and qualification of products and processes, demonstration of innovative services. Among the various areas where the Centre provides results, we can list: Innovative services for telephony, multimedia, data networks, mobile networks, Internet; Evolution of system and networks for fixed, mobile and satellite telecommunication networks, also addressing planning issues; Development of interconnection services and management, to leverage on the infrastructure for services offering; Qualification of equipment, systems and services, quality aspects and environmental impacts for the whole life-cycle of telecom products. CSELT provides also, both to the Group and to others, continuous services for conformance certification to EU in various technical sectors, thanks to its LAP (Accreditated Laboratory).

Achieving Customer Satisfaction through Requirements Understanding

John Elliott

System and Software Engineering Centre,

Defence Evaluation and Research Agency, Malvern, UK

Introduction

One key goal of all businesses is to achieve a continuous and high level of customer satisfaction in the delivery of services and/or products. Such satisfaction is believed to be the basis of long term profitability and business growth. In the sphere of computer based system products, customer satisfaction is dependent on how system development projects evolve to build operational product systems that satisfy the perceived and actual customer need and associated system requirements.

Ultimately, successful customer satisfaction depends upon the depth of 'through-life' understanding about the business need and associated user requirements for a future system, and the ability to communicate those requirements to the system developer. In addition, customer satisfaction and confidence depends upon the level of system assurance offered throughout the system development lifecycle. Requirements understanding problems inevitably lead to poor customer-supplier relationships, unnecessary re-works, and overruns in cost and/or time.

This paper discusses the concepts underpinning customer satisfaction and requirements understanding relevant to software-based system development. In addition, the design of customer-oriented development processes is described together with a process improvement case study and associated experiment. The process improvement experiment was EU project number 23893, REJOICE, whose Final Report [16] can be found at the ESSI VASIE website [17]. The REJOICE experiments and their results have been summarised later in this paper.

Customer Satisfaction, Requirements and Quality

Concept Overview

Customer satisfaction is dependent upon many factors that are associated with the business need, the development project and resultant system product quality. Ultimately the customer is looking for added value to benefit the business operations within a defined timeframe but at an affordable price; hence the customer priority is for an overall *successful business*. The system supplier perspective is to deliver a system within the agreed cost plans to satisfy the customer requirements, thus contributing to the supplier's profit and reputation; hence the supplier priority is for a *successful project*. These different perspectives are typically controlled through inflexible and formal contract management arrangements in the pursuit of a successful project for both customer and supplier. The cornerstone to such *'success'* involves an appropriately rigorous and long-term approach to *'quality'* by customers and suppliers.

This *'success'* discussion implies that customer satisfaction is analogous to overall project success. However, project success, see Garrity [1], also depends on other concepts such as usability and adaptability. *Usability* concerns the wider process use considerations beyond system delivery and acceptance embracing operational experience; this involves different perspectives when applying the new system for individual task support and business organisation performance enhancement. *Adaptability* has a cycle (of planning, doing, filtering and learning) to adjust development progress and direction based on business and development interaction. Both success notions of usability and adaptability are vital to achieve longer-term customer satisfaction.

'Quality' may be loosely inferred to mean 'satisfying requirements' embracing the provision of added capability (i.e. improved business function and performance) and any associated trustworthiness or integrity (i.e. continuously performs as intended without harmful 'side-effects' on business services). One key aspect of the quality perspective concerns the customer and supplier agreeing upon a required level of quality to be achieved within defined and understood cost and time constraints. In addition, the quality level must be defined and be subject to some agreed measurement to monitor attainment.

The remaining development project consideration is the level of *risk* and *uncertainty* associated with the attainment of the required and agreed quality level; the risk perspective depends upon the available knowledge about the project constraints and their implications. Hence both customer and supplier need to understand the level of risk each is taking within their quality level agreement. In practice, the notion of risk sharing between customers and suppliers is a difficult area that influences the nature of any supporting legally binding contractual arrangements. In summary, both customers and suppliers need to plan and implement compatible quality and risk strategies for the development project. These strategies will need to be reflected in any contractual agreements.

Returning to quality within the customer satisfaction arena, customers need to be assured that defined and measurable final *product quality* attributes demonstrate that their defined needs and associated requirements are satisfied. Achieving defined product quality depends upon 'getting the system requirements right' and then 'building the product right' to meet these requirements. This is not easy to achieve especially within traditional contracting processes that tend to encourage the communication of requirements through formal documents and review activities. This inflexible and formal approach to agreement and communication is often the main reason why customer and supplier teams fail to be effective in achieving continuous levels of understanding, which is sometimes coloured by a culture of disrespect and mistrust.

Customer Satisfaction Criteria

The necessary criteria for customer satisfaction are provided below to further demonstrate the relationship with requirements understanding. Such criteria provide the basis for defining measurement schemes from which to systematically argue and justify whether customer needs and requirements have been adequately satisfied.

Area	Criteria
Need and requirements definition and change management	• The business need for supporting necessary or desirable (process and information) change must be clearly defined. • The system requirements must be clear (and error-free) and related to the business need. • There must be an ability to change the product development as the requirements are better understood and refined (or even changed due to business reasons).
Process definition and execution	• The supplier's development process (for all management, engineering and quality activities) must be consistent with best practice. • The supplier processes must closely interface with the customer's processes in executing the acquisition and system creation activities. • The competence and performance of the supplier teams must be of a high standard. • There must be high visibility of the executing development processes and of the product evolution.
Product quality	• The final system product must be compliant with the agreed and understood requirements. • The final system product must meet defined business needs and added value to the customer's business operations. • The final system product must have high levels of usability and be easily integrated into customer processes.
Product management	• There must be sufficient demonstration regarding the satisfaction of business needs, system requirements and product quality (i.e. overall fitness for purpose).

Area	Criteria
	• The agreed project schedule must be met ensuring that the final system delivery and in-service dates are achieved. • The project costs must not be changed without full agreement and justification in customer terms.

Model of Customer Satisfaction and its Components

Partly derived from the criteria above, the customer satisfaction problem domain has four key dimensions, see Figure JJE.1: business need, system requirements, product quality and confidence in quality. This is the basis of a customer satisfaction model.

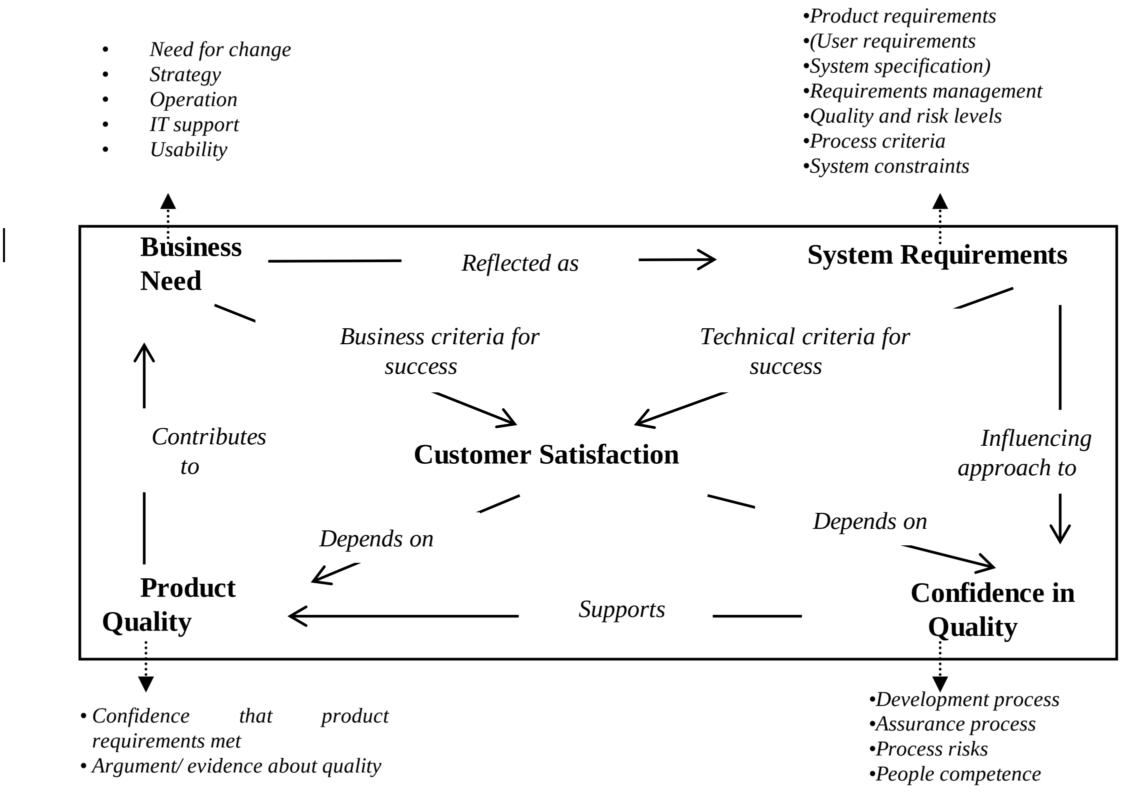


Figure JJE.1 – Four Domains of Customer Satisfaction

Those parts of the customer satisfaction model that address *business need* and *requirements understanding* [12], must ensure that all aspects of user and system requirements are considered. For example, the overall system requirement needs to include the system product requirements as well as those requirements addressing quality and risk levels, development process criteria and the project constraints, e.g. interoperability with existing systems, timescales and costs.

Of key importance to customer satisfaction is the central *product quality* concept, which loosely means 'satisfying the customer requirements or need' throughout the product life cycle, from ‘birth to death’. The product quality requirements will describe a range of external and internal system product attributes; *external* attributes include its functionality and performance (e.g. speed, reliability, maintainability, safety, security, etc) whereas *internal* attributes include its architectural structure, portability etc. Different authors such as Fenton and Gillies [2, 3] describe and review different quality models including that developed for the ISO 9126 standard [4].

The key achievement of actual product quality can only be measured by reference to a quality profile [5] that is a weighted representation of each system product attribute. The attribute weights are derived through customer analyses at the beginning and throughout the development project. The satisfaction of product quality is judged by the combined final 'weighted' attributes achieved against that required through prudent use of project resources to address attributes within designs, trade-offs reviews and their validation.

The product quality achievements depend on the required quality and risk target levels, and the design and execution of development (i.e. creating) and assurance (i.e. checking) processes. This assurance provides the *'confidence in quality'* by demonstrating that the process definition and execution has been effective and demonstrates the required level of quality control. Ultimately, assurance involves checking all levels of the design and provides the argument and supporting evidence, (i.e. as system measurements of 'fit for purpose') that the need and requirements have been addressed to the required quality and risk levels. All processes need to follow consensus best practice that has been suitably tailored to the specific development project needs, while taking into account all associated quality and risk levels. These levels are related to the appropriate process and product criteria. The process criteria reflect the degree of development and assurance rigor to be adopted. The product criteria reflect the design criteria to be adopted in system architectures and detailed design. The customer's confidence in the final system product is affected by the visible degree of thoroughness by which the defined and planned processes were followed and executed; this confidence is also affected by the competence and performance of the development (and customer) teams.

Customer-oriented Lifecycle Processes Attributes

The aim has been to define a technical strategy based upon the fundamental understanding embodied in the customer satisfaction model. The strategy enhances the level of customer satisfaction through improved customer-developer process design with an emphasis on requirements and their understanding. There are three questions to be considered in forming an appropriate technical strategy and in designing a customer-oriented process:

- What are the *attributes* of a customer-oriented lifecycle process?
- How do these attributes relate to current *lifecycle* models?
- What *techniques* are appropriate within a customer-oriented lifecycle process?

Customer-oriented lifecycle process attributes

Based on the concepts described, the following are the key requirements on which to design a new approach to customer satisfaction. The required attributes are below.

Type	Attribute
Frameworks and lifecycles	• *Through-life* treatment of system requirements and business need; this will focus attention on the ultimate project goals and success criteria • Need to embrace the whole system *evolution* lifecycle; this will ensure that systems are not viewed as totally new but rather as add-ons or modifications to existing, albeit larger, systems.
Need and requirements management	• Need to be *flexible* to changing customer needs and perspectives; this will encourage effective contracting and working arrangements to be in place that are based on the premise that such change is inevitable and technical agreements will need to change. • Need to manage the customer needs and requirements and their *satisfaction* through a flexible yet controllable approach to system planning and its execution; this will focus both parties on the theme of customer satisfaction and project success by on-going requirements understanding.
Evolutionary techniques	• Need to ensure that customers get operational systems as a series of *increments* to meet shorter-term priority needs; this will enable customers to get useful employable systems as a series of incremental deliveries formed within an well-founded overarching business system architecture. • Must be *fast* to react to changing customer perspectives about system requirements; this assists customers to quickly see the impact of their desired changes. • Enable executable system prototypes to be *visible* and allowing user 'play back'; this enables the customer team to see the evolving product in concrete terms and respond accordingly. • Need to be able to *roll* the current system solution both forwards and backwards; this assists the speed at which changes (using new or old perspectives) can be played back.
Developer-customer communication	• Need for customer-supplier teams to work in *partnership*; this will enable both parties with separate overall business aims to share a more focused and explicit common project goal within a trusted contractual and working relationship that involves more risk and information sharing, and joint decision making. • Need effective *communication* between customer and supplier teams; this enables a common and shared understanding about the business need, system requirements, and the development processes and products. • Need customers and suppliers to be regularly *interactive* about key business and development changes affecting the partnership; this enables an on-going approach to holism, learning and adaptability throughout system evolution. • Need frequent customer *feedback* to design concepts and system increments prior to final acceptance and in-service use; this will ensure that customers declare timely change based on business use perspectives.
Assurance management	• Need to enable the risk and quality levels to be defined and agreed. • Need to provide effective risk and quality control mechanisms to decide about system *fitness*; this will enable customers and suppliers to understand their shared risk and views about fitness prior to in-service-use.

Customer-oriented processes and current lifecycle models

There is much written about development lifecycle strategies, for example, see Somerville, McConnell and Pressman [6, 7, 8]. The main lifecycle variant labels are: Waterfall; V-model; Spiral; Evolutionary prototyping; Incremental/staged delivery;

Design to schedule; Design to tools; Commercial of the shelf; and Evolutionary delivery. These variants differ in their attempt at imposing different engineering structures for project management purposes based on implicit premises about flexibility and degree of change, speed of delivery, reuse and integration, and system delivery strategies. The overall conclusion is that these lifecycle variants only partially address the above requirements for a customer-oriented lifecycle process and a new approach is required to fully encompass customer orientation. The main lifecycles tend to be sequential, static and prescriptive in nature, and assume all projects need the same process structure. No lifecycle adequately represents the real-world dynamic activities between customer and developer, partly a result of their variability and complexity.

Customer-oriented lifecycle techniques

The major techniques need to support the goals for customer satisfaction and in particular requirements understanding. These techniques cover the following process areas: Business analysis; Communication and interaction; Requirements management and engineering; Project and risk management; Quality assurance; Rapid development; Process assessment, e.g. SPICE and CMM; Project and software measurement; and Object/reuse-oriented design methodologies.
The aim is to populate a customer-oriented lifecycle with a set of relevant techniques, selected from a 'customer-oriented toolkit'. All techniques need to help facilitate the achievement of customer satisfaction and requirements understanding.

Proposed Customer-oriented Lifecycle Processes

The customer-oriented lifecycle processes have been based on a technical strategy that, in turn, has been founded on the customer satisfaction concepts described earlier.

Customer-oriented technical strategy

The proposed strategy is to:

- Define a customer-oriented lifecycle process with the above attributes; that will place an emphasis on through-life ‘requirement understanding’ processes.
- Integrate the proposed lifecycle processes into established project, risk and quality management practices; this will involve identifying the tailoring issues surrounding the introduction of a customer-oriented approach into established software practices and local cultures.
- Propose a set of techniques to support the new lifecycle that is appropriate to a project situation.
- Define a means of measuring the effectiveness of the new lifecycle and supporting techniques in business and project terms; this will focus on the cost-effectiveness using criteria about *identifying need, communication/interaction* and *requirements control*.
- Ensure that the new customer-oriented approach is focusing on business benefits and be widely applicable; this directs the approach to be geared towards the non-

software specialists, needing no specialist tools, knowledge or equipment.

Customer-oriented process overview

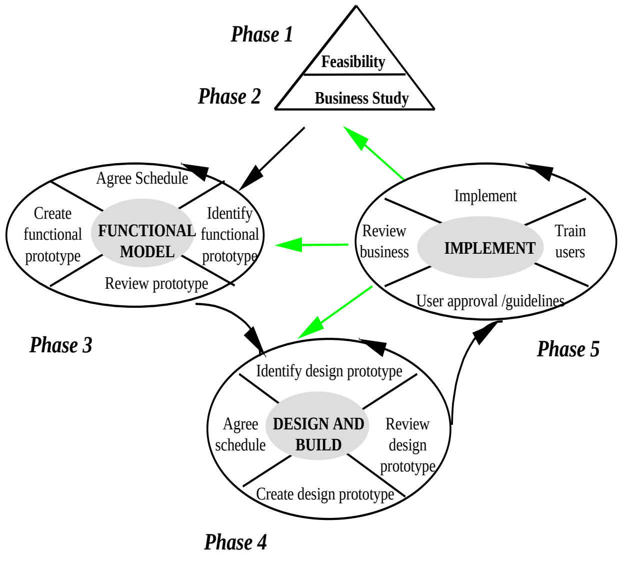
The aim is to establish an improved process and set of techniques that will assist customer and supplier to gain a better understanding of initial and changing requirements so that systems are delivered on time, to cost and actually meeting the customer’s real need. These techniques will also need to address accomplishing and preserving product quality throughout the product life cycle. The approach combines and utilises techniques from separate strands:

- A *customer-oriented lifecycle* process supported by fundamental system models that describe requirements understanding concepts and system 'fitness' measurement.
- Use of *business analysis* techniques such as those exploited in Business Process Re-engineering (BPR) [9] to guide the way in which the customer’s real needs are articulated and understood.
- *Interactive and iterative* approaches such as JAD (Joint Application Development) [10] and RAD (Rapid Application Development) [7, 11] to assist communication and exploration.
- Formalised approaches to capture the statement of *requirements*, support their management and allow traceability, etc.

The customer-oriented lifecycle process has been based on an adaptation [13] of the *Dynamic Systems Development Method* (DSDM) [14, 15] framework. DSDM offers a generic lifecycle framework that is geared to being more flexible, faster reacting and dynamic practices involving joint customer-developer working. Figure JJE.2 shows the five DSDM-based customer-oriented lifecycle process phases. The proposed process adaptations to DSDM, as used within the REJOICE process improvement case study, combine and refine Phases 1 and 2 activities.

Figure JJE.2 - DSDM Based Customer-Oriented Lifecycle Process Framework and Principles

DSDM FRAMEWORK



DSDM PRINCIPLES

I. Active user involvement is imperative.
II. DSDM teams must be empowered to make decisions.
III. The focus is on frequent delivery of products.
IV. Fitness for business purpose is the essential criterion for acceptance of deliverable
V. Iterative and incremental development is necessary to converge on an accurate business solution.
VI. All changes during development are reversible.
VII. Requirements are baselined at a high level.
VIII. Testing is integrated throughout the lifecycle.
IX. A collaborative and co-operative approach between all stakeholders is essential.

The DSDM phases are:

- *Phase 1 - Feasibility Study;* An assessment is made as to whether or not the DSDM approach is correct for the anticipated project. [This is not a conventional form of feasibility, i.e. whether the system concept is achievable.]
- *Phase 2 - Business Study;* Provides the foundations on which all subsequent work is based and provides an understanding of the business and technical constraints. [This study is intended to be relatively short with the aim to describe a 'first-cut' high level requirement.]
- *Phase 3 - Functional model;* this activity is broadly equivalent to a functional specification, but expressed using an executable prototype with some documentation support.
- *Phase 4 - Design and build*; this activity is refining the functionality to reflect non-functional and other quality/integrity requirements; the detailed designs are as executable prototypes but with improving quality attributes, supported by essential documentation.
- *Phase 5 - Implement*: this activity is applying the product within a series of systems trials ultimately being accepted in the operational environment.

The essence of this approach is for the customer and developer to work in partnership ensuring that the needs and requirements are well understood by all. The system is allowed to evolve in terms of refining prototypes resulting in useable increments. The strategy is to be flexible and adaptive to changing requirements and to progressively build quality into the evolving product. The customer-development interactions occur throughout allowing for learning, feedback and adapting to influence development directions. The risk of the flexibility offered needs to be countered through the application of sufficient management and quality assurance practices incorporating process and product checks with sufficient traceable documentation. This approach is to some extent dependent on effective tool-sets in order to gain the customer satisfaction benefits.

Process Improvement Case Study

A case study to examine the effectiveness of the new proposed approach to customer satisfaction and requirements understanding was undertaken as an EU funded process improvement experiment (PIE), referred to as REJOICE, ESSI Project 23893. The purpose of the PIE was to demonstrate whether the new customer-oriented process could provide the business benefits sought as improvement goals.

There are various elements to the experiment:

- Business context.
- Improvement goals.
- Proposed process.
- Experimental considerations.
- Results and assessment.

Business context

The experiment was set in the UK Defence Evaluation and Research Agency's (DERA's) System and Software Engineering Centre (SEC). The SEC is an autonomous development and consultancy business that largely serves the defence system businesses within DERA and the UK Ministry of Defence. The SEC is associated with a very wide range of systems for high technology research, system requirements and design modelling, tool development and operational activities. The SEC operates within a highly controlled business management culture (based on the ISO 9000 series) and its activities are regularly subjected to process assessments (e.g. ISO, CMM, SPICE, EFQM-BEM). The SEC has a 'maturing' software culture supported by its DERA Software Practices. The DERA practices incorporate an in-built measurement system.

Improvement goals

The SEC is striving to achieve the highest levels of CMM maturity (currently achieving Level 3 in some areas) for all its widespread activities supported by the use of SPICE to develop excellence in particular project domains. There were a number of improvement areas identified from various process assessments. This included those concerned with customer relations and ensuring that the SEC met customer needs and requirements. The relevant 'customer-related' goals to be satisfied through an improved approach to requirements understanding were:

- 20% more customer satisfaction.
- No extra effort on requirements activities.
- 15% decrease in requirements generated problem (i.e. less reworks).

Proposed process

The customer-oriented lifecycle process, an adaptation of DSDM as shown in Figure JJE.2, was applied within specific development projects. The adaptation was to combine Phases 1 and 2 of DSDM into a single phase, 'User Requirements Study'. The reason was to remove the DSDM suitability analysis (less important to the REJOICE goals than to rapid application development objectives) and to increase the focus on the feasibility and definition of user requirements against a real, and rigorously studied, strategic need for business change. Hence, this new phase focuses on the communication, understanding, elicitation and high level capture of business needs and requirements. In addition, before the adapted DSDM lifecycle process (referred to as the *REJOICE process*) can be applied, further DSDM 'tailoring' considerations need to be addressed:

- How can the flexible proposed process be utilised within a high-control business and quality management culture?
- What standardisation process details should be defined and to what level of detail?
- How do you define the exact process incorporating methods and tools to apply to a specific project?

It should be stressed that the new customer-oriented process represents a major shift in development culture, a major issue for the REJOICE experiment. In support of the new process, a set of specific methods and tools were selected from which the experiment process details were selected. There was an emphasis on business analysis (e.g. BPR), interaction management and facilitation (e.g. JAD), design methodology (e.g. object-orientation) and requirements management support (e.g. procedures and tools).

Experimental considerations

The experimentation was divided into four parts:

- Experiment 1 - Defining, tailoring and introducing the new customer-oriented 'REJOICE' process.
- Experiment 2 – Partial Application of the REJOICE process to the development of a Requirements Modelling Tool.
- Experiment 3 - Applying and measuring the impact of the 'REJOICE' process during the development of a DERA Intranet based CMM Self-Assessment Tool.
- Experiment 4 - Comparing the 'REJOICE' process with the existing development process during the development of a DERA Intranet based CMM Self-Assessment Tool.

Each experiment had its own design that included a number of specific hypotheses to be tested and an associated measurement scheme, each of which was linked to the improvement goals. Overall the measurement strategy included maximising the use of qualitative observations backed up by argument based on valuable experience identifying the issues, in addition to collecting quantitative measures. The data collection involved a combination of surveys, interviews, project resource extracts and tracking what processes were being implemented in some measurable detail. The major experimental part was the application of the new process to be applied to two tool development projects. Each project had specific and well-informed customer teams; one project was a requirement modelling tool and the other was a CMM assessment support tool. The outline measurement scheme to examine the new process is shown below (more details are described in [16]).

Goal	Area/Factor	Metrics:
20% increase in satisfying customer needs	***Customer Satisfaction:*** meet need; confidence in product; confidence in process/people ***Product Effectiveness:*** product quality claimed; demonstration of quality	Satisfaction (score) with project, product, process, people No. of prototype releases - planned, actual No. of the original satisfied/unsatisfied requirements No of requirements changed No of requirements priority changes No of evolution's of requirements

No <u>change</u> in costs of requirements activity	***Project efficiency:***
	process definition;
	process cost;
	people impact
		Time spent in customer interactions
		Number of customer interactions
		Time spent demonstrating models/prototypes

15% decrease in problems due to poor requirements understanding	***Project efficiency:***
	requirement defects;
	people interaction;
	process cost impact
		Number of requirements not satisfied
		Effort spent satisfying incorrect requirements

Goal	Area/Factor	Metrics:

Experimental results

The main results of the four experiments are detailed in the REJOICE Final Report [16] that provides detailed qualitative and quantitative (measurements) evidence presented in a form that argues about the validity of the various customer-oriented process hypotheses. The overall results are now briefly summarised in the following table.

Experiment	Main Results:
Experiment 1 Defining, tailoring and introducing the customer-oriented process.	• Successive levels of tailoring are involved - they are difficult to clearly define • The DSDM based customer oriented framework is 'loosely' defined and requires further refinement and instantiation to be employable • The new DSDM based process does not fit easily with existing Quality Systems • Detailed DSDM based processes cannot be fully prescribed due to the highly iterative processes involved that is dependent on actual product development progress • Detailed project planning cannot be achieved: plans need to stay at a high level or they will lag behind the actual development
Experiment 2 Applying and measuring the impact of the new customer-oriented process: Requirement Tool Project	• The pragmatic use of principles leads to a 'fit for purpose' product • 'High level' user requirements are difficult to resolve and manage contractually • The use of prototyping techniques are very effective • The contract requirements would not have been met if traditional processes used
Experiment 3 Applying and measuring the impact of the new customer-oriented 'REJOICE' process: CMM Assessment Tool Project	• There was good 'buy-in' by the development team • There were high levels of user involvement • There was a high level of user satisfaction with the final product • The users sometimes resented the demands on their time • The team emphasis on development of product means documentation/testing suffers unless control exercised; this may be a problem for longer term customer satisfaction • Any organisation and culture changes are non-trivial • It was difficult to control and plan prototyping • It was difficult to monitor project progress with traditional management techniques • The development team was not used to empowerment and they tended to perceive a lack of direction and management
Experiment 4 Comparing the new customer-oriented 'REJOICE' process with the existing traditional development process.	• It is difficult to compare results with 'traditional' methods due to non-equivalence with stages in 'waterfall' and variants. • Customer surveys provided evidence of improved satisfaction • The REJOICE process was found to be more efficient than traditional methods in terms of required functionality achieved for developer effort • If the development had followed the existing traditional process, that may have led to the development of an altogether different tool, not taking

Experiment	Main Results:
	into account real business need • The longer term customer satisfaction advantages are more difficult to assess • The REJOICE process developed products may be more difficult to maintain and evolve

The collective evidence from all these experiments provides the basis for deriving the lessons that have been learnt within the REJOICE process improvement case study in terms of the technological and business impact of the new DSDM-based REJOICE process. As in the REJOICE Final Report [16], these lessons are now described in terms of these technological and business viewpoints.

Lesson Learnt

Lessons learnt - technological viewpoint

This viewpoint assesses the impact of the new process in relation to current software practices and their evolution. The lessons are:

- Adoption by the SEC of a new, evolutionary yet controlled lifecycle approach (where appropriate to the projects) is expected to lead to improved customer satisfaction.
- DSDM offers a useful set of concepts (sensible principles, flexible requirements philosophy, strong user and end product focus) that will advance the SEC best practices.
- DSDM is not only suitable for 'RAD type' projects but its concepts can be integrated, in full or in part, into more traditional lifecycle approaches.
- The integration of the DSDM based process within a traditional ISO 9000 quality controlled software development operation is non-trivial, unless DSDM is used to do RAD developments only.
- Commonly available tools generally support the basic DSDM based REJOICE process although more model based tools are needed that facilitate effective user-modelling interaction (to study requirements and acceptance testing issues).

Overall, the technological lessons about the DSDM based REJOICE processes are fundamental. More radical software lifecycles are designed to improve customer-developer relations. These require new ways of thinking about project control and tool based cultures. There is clear evidence that the REJOICE process is sufficiently mature and does indeed enhance customer satisfaction, assuming that a joint product-focused management approach is taken by both customers and developers. In short, the REJOICE process offers clear claimed benefits when used in part or in full, but there are a number of non-trivial project and quality management issues to overcome.

Lessons learnt - business viewpoint

This viewpoint assesses the impact of the new process in relation to business goals and activities. The lessons are:

- Customer satisfaction and the attendant advantages are likely to be achieved by the using the DSDM based REJOICE Process.
- The REJOICE process is likely to provide cost saving gains in the efficiency of requirements-based activities, dependant on project complexity and associated implementation issues.
- The REJOICE process requires a co-operative product focused management approach.
- Definition and management of contractual boundaries will be challenging.
- Cultural changes may be difficult to manage.
- Consider applying DSDM techniques to smaller projects until confidence is gained.
- A REJOICE type process will increase business opportunities through improved customer relations.

Overall, many software businesses, often Small Medium Enterprises, should benefit from the DSDM-based REJOICE concepts, process and techniques in terms of customer satisfaction and requirements efficiencies. However, the degree of success will depend upon the organisation and customer culture, the appropriate application to suitably complex projects and an effective use of available software technologies. In short, the REJOICE process framework is well founded but its success critically depends on the management of people and technical resources during any development project implementation.

Summary

This paper has described the underpinnings and development of a customer and requirements focused ‘REJOICE’ process that has been adopted from DSDM. The underpinning arises from the evolving development of an innovative customer satisfaction and requirements understanding model that has a key system measurement component. A new customer oriented lifecycle process has been defined and examined within an EU funded process improvement experiment, REJOICE. REJOICE has focused on the business impact of a requirements-oriented process improvement geared to improve customer satisfaction; the business goals include improved customer satisfaction and cost effective requirements management. The experimental findings support the main hypothesis that the flexible process should yield the business benefits suggested; however a careful approach to process introduction is required as a new cultural approach to customer-supplier partnerships is critical. If implemented well, both customers and suppliers should reap major benefits.

Acknowledgements and Disclaimers

The author would like to acknowledge the European Commission for funding the Process Improvement Experiment, REJOICE, Project 23893, supplemented by internal Defence Evaluation and Research Agency funding on the general concepts underpinning requirements understanding. Furthermore, the author would like to thank the dedication of the REJOICE team, in particular that of Peter Raynor-Smith for his major contribution.

The views expressed in this paper are entirely those of the author and do not represent the views, policy or understanding of any other person or official body. Further details can be requested from the Defence Evaluation and Research Agency (DERA Malvern), Systems and Software Engineering Centre, Tel: +44 1684-895161, E-Mail: jjelliott@dera.gov.uk.

References

[1] Garrity, E.J., Saunders, G.L., "Information Systems Success Measurement", IDEA Group, 1998.

[2] Fenton, N. E., Pfleeger S. L., "Software Metrics", 2nd Ed, Thomson Computer Press, 1997.

[3] Gillies, A., "Software Quality - Theory and Management", Chapman and Hall, 1992.

[4] ISO 9126, "Software Product Evaluation", 1992.

[5] Van Ekris, J., "Towards business oriented questionnaires for the specification of software product quality", ESCOM-ENCRESS 1998 Proceedings, May 1998, pp230-238.

[6] Somerville, I., "Software Engineering", Addison Welsey, 1996.

[7] McConnell, S., "Rapid Development", Microsoft Press, 1996.

[8] Pressman, R. S., "Software Engineering - A Practitioner’s Approach", McGraw Hill, 1997.

[9] MacDonald, J., "Understanding Business Process Re-engineering", Hodder & Stoughton, 1995.

[10] Bell, S., Wood-Harper, T., "Rapid Information Systems Development - System Development in an Imperfect World", Second Ed., McGraw-Hill, 1998.

[11] Martin J., "Rapid Application Development", New York: Macmillian, 1991.

[12] Elliott, J. J., "System Understanding Reference Model", DERA Report, 1999.

[13] Raynor-Smith, P. M., "REJOICE Process", DERA Report, 1998.

[14] DSDM Consortium , "DSDM Manual", 1996.

[15] Stapleton J., "Dynamic Systems Development Method", Addison-Wesley, 1997.

[16] Raynor-Smith, P. M., Elliott J. J., REJOICE Final Report, Version 1.0, May 1999.

[17] ESSI VASIE website: http:// www.cordis.lu/esprit/src/stessi.htm

Session 5
SPI and Establishment of Models/Processes I

Chairman

Tor Stalhane

Sintef, Trondheim, Norway

Process Description and Training: The Two Sides of the SPI?

Janis Plume
Riga Information Technology Institute
Kuldigas iela 45, Riga LV-1083, Latvia
E-mail: janis.plume@dati.lv

Introduction

This paper presents a summary of experience that has been gained at the DATI organization during activities aimed at process improvement. DATI is the largest contract software developer in the Baltic States, and process improvement activities are the focus of considerable attention in the organization, which employs 400 software engineers. Process improvement activities at DATI are the area of operations of a subsidiary, the Riga Information Technology Institute (RITI).

The approach to process improvement that is used by the organization corresponds to recommendations by such authors as Zahran [1]. The three main phases in process improvement are description, training and enforcement. In this paper, I will devote particular attention to the description and training phases. Experience shows that a reduction in the level of detail of a process description can be a sensible procedure, as long as good process improvement results are obtained by strengthening the respective training program.

Thus, the approach that has been accepted at DATI is aimed at ensuring that process descriptions contain only the most important process activities. We believe that the training of our personnel is adequate to allow us to tailor standard processes to our project needs, supplementing them with appropriate supporting activities in-house.

The establishment of a solid training program is of great importance in knowledge-intensive production processes, especially when we are talking of software engineering – a sector in which technologies (tools, methods) [10] are developing with enormous speed. The new technologies in and of themselves have an effect on development processes and dictate changes therein. Process changes can also be initiated within the organization itself.

The advantage of a training program as compared to process description is that there are direct links between the designer of a process (the instructor or teacher) and the user of the process. The program can be used to discover the level of understanding among employees with respect to the various processes, and arguments can be developed for the initial changes that are needed.

There is, however, one clearly visible shortcoming when emphasizing training in the area of process improvement. When staff turnover occurs, the performance level of an organization declines, and this means that additional attention must be devoted to ensuring that employees are motivated to stay with the organization. It must be clearly understood that when a staff member leaves, the organization loses critical resources that have been spent on the training of that particular individual.

The main goal of this paper is to analyze the costs of the process description and training phases of a process improvement program, so as to demonstrate the links between the costs of the two phases.

Process description

The DATI organization has established a process improvement group to handle process description. It is responsible for coordinating improvement work, as well as for tending to and maintaining process descriptions. This is the main infrastructure element in the area of process improvement, and management representatives from DATI are involved in the process.

The selected approach

The process description approach is based on traditional considerations:

- Standards which set out process classification;
- Process notation which is used to describe the sequence of specific elements in the process.

It is up to the process improvement group to select process classifications and notations (language) which are used to provide process descriptions. The basis for the process classification is the IEEE 12207.0 standard [5], or the ISO/IEC 12207 standard (these are basically equivalent).

The most important principle that emanates from these standards is the division of processes into categories – primary, supporting and organizational. The highest priority is attached to primary and supporting process groups. From the very beginning in the description phase, it is clearly understood that each project process description contains a specific combination of primary and supporting processes.

The main reason for the simplified process description is the specific nature of contract software development, because the processes in various projects can differ to a great extent, depending on the client with which work is being done.

There are several arguments which underpin the reduction in detail when it comes to process descriptions:

1. As the level of detail in process descriptions increases, the descriptions become massive, hard to maintain and difficult to understand;
2. The performance of supporting activities in a project involving contract software development is often based on an agreement with the client, not on specifications from the organizational process.

The notation was chosen to be as simple as possible, based on the following considerations:

1. The process description must not contain supporting activities which complicate the process description, make it cyclical, or establish branching;
2. The process description is not based on the development of a concrete product. The process description is not aimed at a concrete result, because the result of one and the same process can differ in various projects, both in terms of form and in terms of content.

Thus, the notation in the process description is reduced to a simple chain of activities, in which each activity is given entry information and exit information. There are two kinds of entry information in an activity:

- Quality management system templates, forms, instructions, etc.;
- Information emanating from within the project (the concept "information" here is used in the same meaning as the concept "life cycle data" is stated in the IEEE 12207.1 standard [6]).

A few thoughts about the concept of project information. Information that comes from processes and activities in an actual project is distributed via documents or products. The grouping of such information is up to the project handlers, and it is a part of tailoring the respective process. The distribution of information among documents is usually based on the standards which are being used as the foundation for the project work. This is something that requires agreement with the client, and here, too, it is assumed that the project personnel have sufficient skills to do the necessary tailoring.

Training in the organization

Training in the organization is critically important when it comes to process improvements, and this in several ways.

First of all, the effectiveness of an untrained process is significantly lower than that of a trained process, irrespective of how precisely the process has been documented. By untrained process here is meant a process, which users are not trained in performing this process. This assumption was a cornerstone of our training program and corresponds to the ideas of [1].

Let us stipulate that the mission in training is to ensure that employees of the organization understand the defined processes and are able to carry them out. A well-trained employee is one who is familiar with the process description and can interpret the description and tailor it to the respective project situation. It's also important to understand that personnel must be able to react to changes in the process definition and to work in accordance with those changes. The main benefit of a proper training program is the fact that when small process changes occur, employees find themselves able to work in concert with the changes, without any need for additional training.

Secondly, training programs must deal with the problem of resistance among personnel – a problem that is not denied by a variety of authors, including [8] and [2].

It is assumed that process improvement must be a democratic process, but a democratic approach to process improvement does not resolve the problem that attitudes toward such activities tend to be negative. It has been found that many employees consider participation in democratic processes to be a burden that keeps them from their everyday work. Participation in process improvement activities requires additional communication skills among employees, and software engineers are often not prepared for this. As a result, those who initiate process improvement activities are often seen as wasters of money who are not handling the day-to-day problems of employees. The work of process improvement groups in this kind of atmosphere is less than effective.

This means that a second – and no less important – mission for training is to overcome personnel resistance. When training courses are organized, groups of specialists from various sectors are set up, and it is within these groups that an environment is formed where participants can reveal the weak points and shortcomings in the processes that are being elaborated in a timely way, and where there are favorable conditions for communication.

The significance of a quality manual

An important element in maintaining skills that have been learned in training programs is the quality manual of the organization [4]. A quality manual for our purposes is a brief and general description of the activities that occur within an organization, not a detailed description of the organization's software development process.

A process description, therefore, consists of two levels:

- The level of the quality manual, which describes the accepted practices in the organization or, to put it another way, the culture that has been accepted in the organization;
- The standard process description level, which consists of instructions, check lists, etc., as well as from more traditional and standard charts of the processes, which can be used in the projects as larger fragments.

The intermediate level, which differs from project to project, does not contain a concrete description, and its implementation under the auspices of projects depends on the knowledge and skills of personnel, through which the standard process description is tailored to the project (see Fig. JPL. 1).

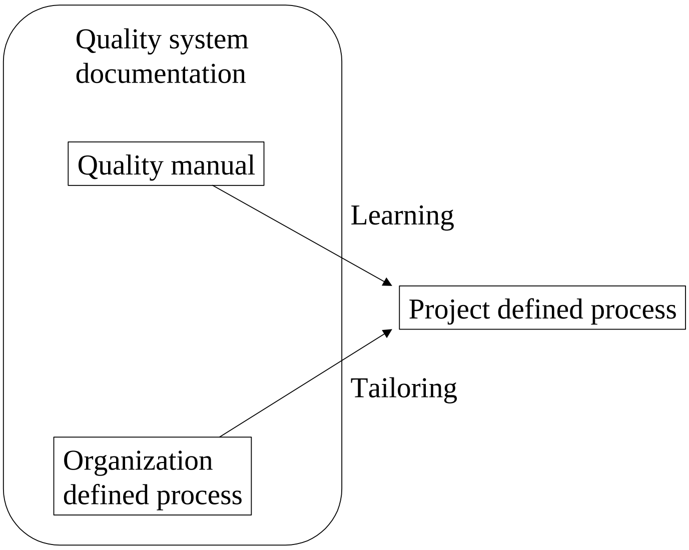


Figure JPL.1. The quality system and the project

In order to make it easier to reach the goals of a training program – especially the goal which has to do with the ability of personnel to use process descriptions – the training program must be based on the principles that are set out in the quality manual.
Thus, the content of the training is more in line with what is described in the quality handbook, also addressing the way in which the process descriptions are to be used in order to tailor them to the project specifics (the tailoring guidelines).
In short, we can say that the role of training is to establish a certain culture in the organization, and it is the role of the quality manual to serve as a description of that culture.

The training program

Another critically important element in the training that occurs within an organization is teaching employees about the new technologies and methods that are appearing in the world all the time and in great volume. This is important irrespective of the work that the organization does. Training of this kind will not be discussed in detail in this paper, however. Here we are addressing training that has to do with handling the processes of an organization – the information that has been created within the organization.
Process training is in and of itself a complicated process, for several reasons:

- It is hard to find process specialists who have the proper pedagogical skills and understanding of the organization’s SPI goals;
- Training may show no immediate results. The cost of this may be a loss of management support for a training program as one of the key elements of process improvement activities. Without management support any activity can be doomed to failure. In the next part of the paper I shall propose a solution which can be applied to this problem.

During the course of training programs, it is important to show that there are several ways to carry out supporting activities in a primary process. In other words, the tactic in a training program should be to demonstrate the link between the quality manual and real-life projects. This is easiest to do with the help of examples, which means that instructors in the program must be highly experienced specialists from the respective process group.
The process of developing a training program involves the following:

- Selecting and appointing the process instructors;
- Preparation of the training materials;
- Planning of personnel training;
- Staging of the training courses in accordance with the plan;
- Evaluation of the results of the training.

Training courses must be mandatory for all of the employees in the respective organization’s production process.

Measurements

A measurement program is used to evaluate the effects of training [9]. The indicators and figures in this part of the paper do not reflect precise measurement results, and they can subsequently been updated. The figures are based on the early results of a measurement program that is still being conducted, and in some instances they represent hypotheses that will be checked in the future but which have obtained trustworthy justifications during initial operations.

Effort

With respect to the effort that is needed in process improvement, it is taken into account that the costs associated with each process improvement activity emanate from the initial implementation of the activity and its maintenance in the case of changes. It is assumed that the basic costs exist in the maintenance area, because the most important thing is the long-term evaluation of process improvement costs.

In this context, the mission of process description is to maintain the timeliness of process descriptions, taking into account any changes which occur as the result of the introduction of new technologies and other ways of optimizing the process.

The mission of training programs in the maintenance phase is much more complicated – to maintain the level of skills in process applications in the organization. The effort needed in the training program is influenced not only by changes in processes, but also by the speed of staff turnover. Here we are speaking not only of the hiring of new workers to replace departed ones, but also the retraining of personnel within the organization.

Let us assume at this point that the three main indicators describing the effort that is needed in process description are:

- The complexity of the process description. This indicator actually consists of two parts – the complexity of the process description's notation or language [7], and the level of detail in the process description. The simplest methods for process description are those which require no other skills than the ability to read. Notations which are the basis of modern business modeling tools can be seen as complicated, because they involve quite a few (up to 10) different concepts (symbols). The level of detail in process description is closely linked to the complexity of notations or language. There is no point in using primitive description language in highly detailed phase descriptions, nor is it useful to use complicated language when describing a general process. The complexity of a process description as a purely quantitative measurement is something that must be studied in greater detail;
- The size of process changes. Here it is rather simple to select a purely quantitative indicator. On the basis of the selected process description notation, it is possible to classify the elements that are used in the description and to count them precisely. In that case, when changes occur, it is possible to evaluate how much of the previous version of the process description must be changed (added, deleted);
- Personnel turnover rate. This is a traditional indicator which shows the extent to which employees of the organization feel motivated to work at the organization and to be loyal to it.

Comparisons of the necessary effort are shown in graphic form in further sections of the paper.

Process description

The effort involved in process description is directly dictated by these two indicators:

- The complexity of the process description notation and the level of detail in the process description itself, i.e., the two indicators that underpin the whole concept of process complexity. In the graph that is shown in the next part of the paper (Fig. JPL. 2), it is assumed that effort increases in a linear way. There is, however, a yet-untested hypothesis which posits that as complexity increases significantly, the level of effort may increase even more.
- The size of process changes. Here, too, it is assumed that effort increases in a linear way (Fig. JPL. 4). Changes in the process occur via traditional change management techniques [3].

When both of these indicators are combined, the effort needed in process description can increase considerably, even to the point where the initially allocated resources for process description are no longer adequate for the maintenance of those descriptions – collecting requests for changes and implementing the changes in descriptions which have already been decided upon.

It is safe to say that the effort involved in process description is in no way affected by personnel turnover rate.

Training

The costs of maintaining a level of skills have to do with two major indicators:

- Personnel turnover rate, which affect training costs in the most direct way and are, in fact, the most important indicator to be taken into account in an organization that emphasizes training in process improvement activities (Fig. JPL. 3);
- The volume of changes. As was noted previously, process changes can occur for two objective reasons – the appearance of new technologies and improvement activities which are aimed at process optimization. It is undeniable that there are costs associated with the implementation of a new process description, but in the case of small process changes, there is no need to gear up a training program. Personnel adapt easily to small changes in the process. If previous skills have been sufficiently convincing, no new training processes are, in most instances, needed. It is enough to conduct an information campaign about the fact that changes have been made to the process, and that is what makes up the cost of implementing the new processes. If, however, the changes are sufficiently significant, it may well be that training is necessary. In that case there is a large jump in expenditures (Fig. JPL. 4.).

Our approach to organizing training programs (see the section on training in the organization) makes the training process less dependent on the complexity of the process description that is being taught. It is not the process changes that create additional effort in the training process, because training programs are largely based on the quality manual, and the main goal is to train personnel in the basic principles of tailoring.

Comparative graphs

The following figures show qualitative comparisons of the effort that is needed in process improvement activities. The graphs show the influence of one indicator on the effort needed for training and description.

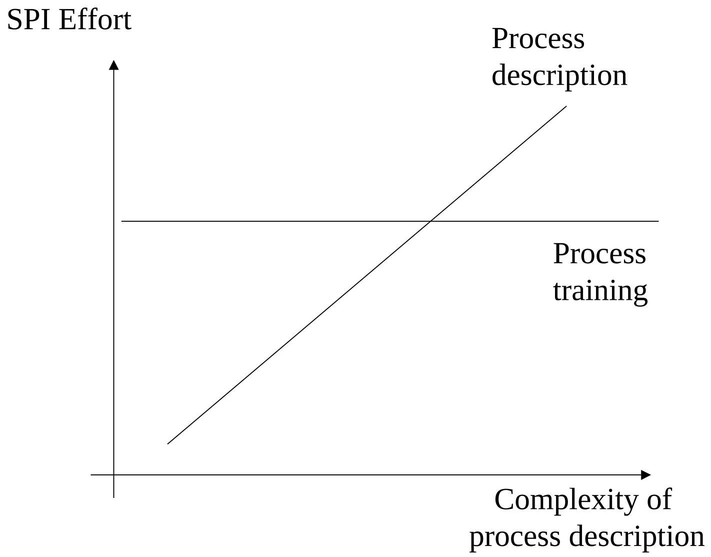


Figure JPL.2. SPI effort and process description complexity

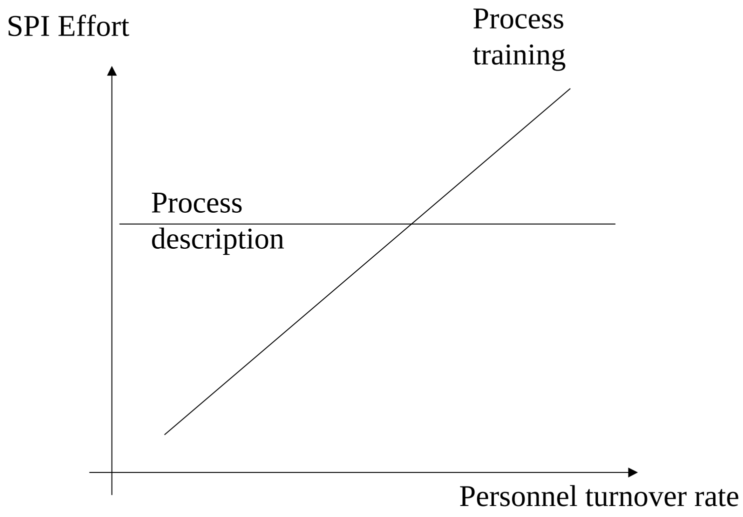


Figure JPL.3. SPI effort and personnel turnover rate

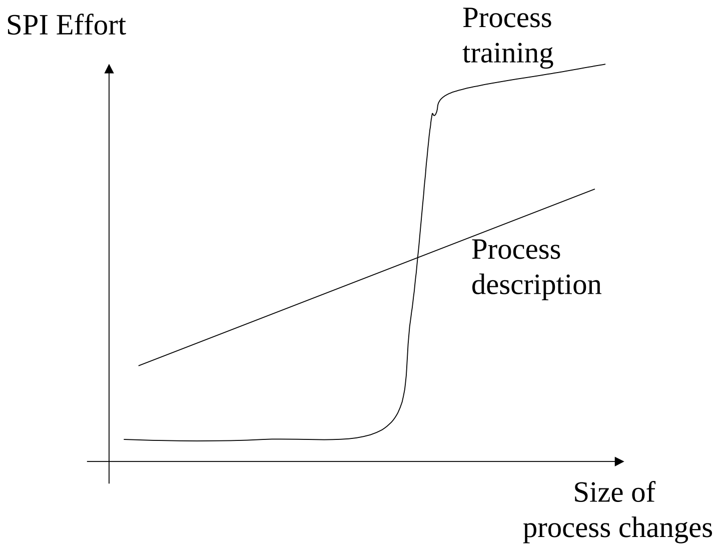


Figure JPL.4. SPI effort and process changes

Inconsistencies in the process

In order to conduct a full cost-benefit analysis, it was necessary to elaborate a measurement whereby we could specify the effectiveness of the process implementation (in this case – description and training). The basic approach is a measurement of improvements in the performance of the process, as well as a decline in the density of defects in the results that are obtained through the process. If we look at the direct goals of training activities, however – the fact that personnel must be made to understand the various processes – we can use an indirect measurement that is universal for several process groups.

We decided to measure process effectiveness by counting the number of clear inconsistencies in the process instances. In other words, the issue is the extent to which the defined process is being followed. This level can be measured in several ways. One way which would allow us to judge whether the process is being implemented correctly, would be counting the number of inconsistencies in that type of process. The collection of such data, however, is too complex – indeed, how are we to define an instance inconsistency? A fairly close approximation, therefore, is the relationship between process instances and incorrectly implemented processes. In this case the number of process instances must be sufficiently high to allow the measurements to illustrate general trends.

In this case each process instance must be reviewed only in the context of asking whether it is being implemented correctly or not. This measurement, however, is also not without its problems. It requires a regular and fairly frequent review of a certain cohort of processes. Such measurements are a natural result of the next phase of process improvement – the process enforcement phase.

The use of a measurement during the implementation of process improvement activities is shown in Fig. JPL. 5. The number of instances of non-conformity is not stable – as is the case in the software maintenance phase, the improving changes cause a varied occurrence of problem reports.

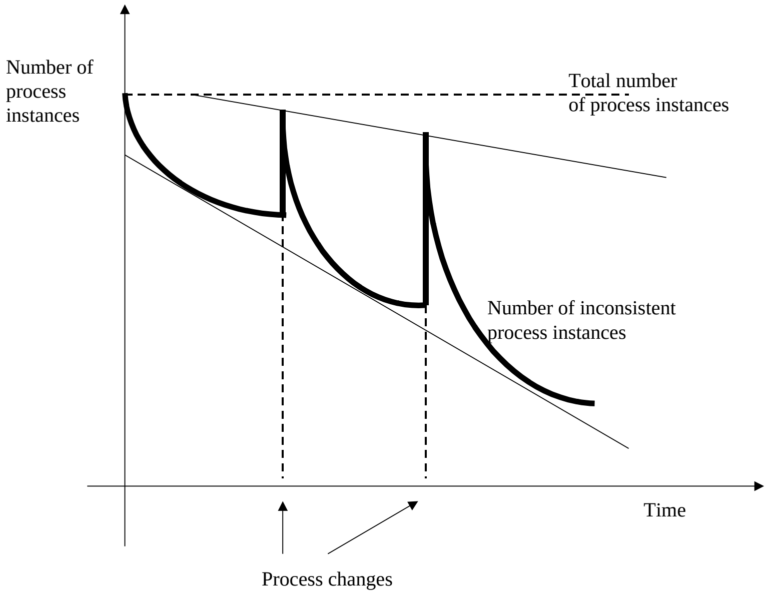


Figure JPL.5. Number of inconsistent process instances

The result of an effective training program is a more rapid reduction in process inconsistencies. In Fig. JPL. 5 this is represented by the line which limits the number of inconsistent process instances from the bottom. This means that the process users are themselves adapting the process changes, and the changes become less painful as a result.

This measurement can be used to demonstrate to management the investment of process training in an overall process improvement program. This is particularly effective if the same measurement is applied to an untrained process, and then the results are compared.

It must be taken into account that improvements in these indicators can be interpreted in various ways, which means that the reason for a process inconsistency must also be considered. There are three main reasons why inconsistencies occur:

1. The process is not understood correctly;
2. Employees are lazy or negligent;
3. There is a crisis situation.

Training can battle only the first type of inconsistency. The fight against the second problem is an issue for process enforcement, while the avoidance of crisis situations has to do with the overall position taken by an organization with respect to the issue of whether key departures from standard processes are to be permitted or not.

Conclusions

These are the main conclusions that can be drawn from the current phase of the process improvement program that is being implemented at DATI:

1. An important measurement in evaluating process effectiveness is the number of process inconsistencies that are revealed. The effectiveness of training is indicated by the ongoing reduction in this indicator, especially when the reduction is occurring more rapidly than in a process where there has been no training. This indicator tends to be highly variable, appearing in waves on a chart. The high points correspond to the time element of changes conducted in the process. It is critical to conduct such measurements so as to prove to management that training programs are beneficial. A diagram of a trained process will show a more rapid decline in the number of inconsistencies than that of an untrained process.
2. There is good reason to encumber a training process with additional functions that are not particularly traditional in training as such. A training process involves many additional opportunities which, when sensibly used, can improve training results fundamentally, and at little additional expense. This is most true with respect to the ability of staff to use described processes. Training sessions also serve to reduce resistance among personnel and dislike against process improvement activities as a whole. These training functions must be clearly understood, and the training program must be set up in accordance with these functions.
3. It may at first seem that training-oriented process improvements are more expensive, and greater investments are needed to improve the process. In practice, however, it is the training phase in which process improvement resources are saved. It is frequently believed that a process description is adequate to allow employees to train themselves in the use of the process. A recommended approach, however, is one in which employee training is always part of any process improvement activity. In that case it makes sense to save effort on the process description phase.

The main way to preserve training-oriented SPI effectiveness is to provide for personnel stability and to motivate employees to work in your organization. Emphasis on training is in line with the idea that in a knowledge-intensive area of production which changes frequently, training in the organization can make all the difference.

Future research could focus on the process enforcement phase, seeking to specify the burden that is placed on this phase as a result of investments, or lack thereof, in the first and second (description and training) phases. Control processes will not work if necessary training is not done. In such instances the resources needed for control increase considerably, and the result is that control functions are not implemented, and the actual implementation and supervision of processes are lost.

References

[1] Zahran S. *Software Process Improvement – Practical Guidelines for Business Success,* Addison-Wesley, 1997

[2] Humphrey W. *A Discpiline for Software Engineering,* Addison-Wesley, 1995

[3] Pressman R. *Software engineering- a Practitioner's Approach,* McGraw-Hill, 1997

[4] Ince D. *Software quality assurance – a Student Introduction*, McGraw-Hill, 1995

[5] IEEE/EIA 12207.0 *Industry implementation of International Standard ISO/IEC 1207:1995. Software Life Cycle Processes.*

[6] IEEE/EIA 12207.1 *Guide for Information Technology. Software Life Cycle Processes. Life Cycle Data*.

[7] Martin J., McClure C. *Diagramming Techniques for Analysts and Programmers*, Prentice-Hall, 1985

[8] McFeeley B. *IDEAL: A User's Guide for Software Process Improvement*, Software Engineering Institute, 1996

[9] Solingen van R., Berghout E. *The Goal/Question/Metric Method – A Practical Guide for Quality Improvement of Software Development*, McGraw-Hill, 1999

[10] Solvberg A., Contents of an Information Systems Engineering Education, in: *Proceedings of the Third International Baltic Workshop on Databases and Information Systems*, Vol. 1, pp. 3-6, Riga, 1998

The impact of a new software development methodology and how to afford the changing resistance: the RESPECT experience

E. Cozzio
Federazione Trentina delle Cooperative,
Via Segantini, 10 - I-38100 Trento, Italy
tel. +39 0461 898320, fax +39 0461 895431
e-mail: enrico.cozzio@ftcoop.itetc.

F. Calzolari
ITC-Irst, I-38050 Povo (Trento), Italy
tel. +39 0461 314583, fax +39 0461 314591
e-mail: calzolar@irst.itc.it

Abstract

The software industry has to cope with the rapid technological evolution and the global market competition, in order to satisfy the growing user demands for quality, services and bandwidth.

Although experience in developing systems has shown that an inadequate understanding of system requirements is the single most important cause of user dissatisfaction and system failure, the software development process is often largely unformalised and it lacks of support for the early phases of requirements collecting and definition, especially in small companies.

Therefore the FTC (FTC stands for the Trentino Federation of Cooperatives) addresses this problem, providing a way to move from an informal and unsupported software development process to a more formal one, adopting new methodologies and applying suitable tools. The main technical objective of the Process Improvement Experiment RESPECT is to improve the requirements' specification and analysis phase by formalising the process of requirement capturing and by adopting a CASE tool to support this phase.

Unfortunately one may expect that moving from an informal development process to a more structured and formal one will add an overhead to the programmers' activities. However, from the business point of view the challenge is to measure the impact of process changes in order to assess that the new practice really improve products' quality, time to market and customer satisfaction at the price of some added burden for development activities.

This article summarises the first year experience with the RESPECT project and addresses the problem of measure the impact of the new development methodology and how FTC afforded the changing resistance in order to make the improvement really effective.

Introduction

Federazione Trentina delle Cooperative s.c.r.l. (FTC) is a non-profit association, organised as a consortium of co-operatives operating in the area of Trento, a small town in the heart of the Italian Alps. It provides a set of services for all its members, with the objective of maximising synergies and implementing a uniform set of standards in the region (see Appendix 2).

ICT support for the whole of FTC is provided by an internal software development team, the responsibilities of which span from software development to maintenance, from on site assistance to network support.

In recent years the FTC has identified a potential expansion of its overall activities from the captive market of the represented co-operatives to the external market. This poses a set of new challenges for the software development team, in particular regarding the way expectations of external customers are addressed.

This is primarily due to the fact that, as it happens with many small companies, FTC' software development process is largely informal and deadline driven. In this context, the capture of user requirements is usually done in an *ad hoc* and often imprecise fashion, with the assumption that the real expectations will be clarified along the way through informal progress meetings. This has proven wrong in several occasions and is one of the long-standing identified weaknesses of FTC' development process.

In the awe of the business expansion for FTC, the software development team has the opportunity as well as the duty to address the issue in a systematic fashion. FTC' software development team has therefore initiated an improvement programme for the entire software development process, starting from the requirement specification and analysis phases.

The RESPECT project represents the starting point of this programme. The aim of the experiment is to implement a new requirement specifications process supported by automatic tools, and to show that improvements in the requirements specifications process enable FTC' software staff to decrease the overall development effort and increase software quality.

If the experiment will be successfully completed, two primary objectives should be reached:

- FTC' management will have evidence of business benefits deriving from software process improvement and will be able to take informed decisions regarding future improvement actions;
- ICT staff will be more inclined to adopt new methods for software development and will be more willing to accept the overheads in exchange for reduced overall effort.

For this experiment FTC is being assisted by ITC-Irst, a public research institute whose activities spans from microelectronics to software engineering and software maintenace. ITC-Irst co-operates with the FTC for scientific and methodological aspects, supporting activities regarding tool selection, customisation and training, implementation as well as requirement process and guidelines definition.

The starting scenario

The software development process early followed by FTC programmers is typically iterative. A prototype is usually developed to gain feedback from the user, but not all of the required features are immediately implemented into the prototype. For each iteration, only a fraction of the requirements are implemented. When a meaningful subset of all the requirements is implemented into the currently developed prototype, a working (sub-) system is delivered.

Although the software process has not yet been formalised nor has a standard methodology been defined to describe the nature of each phase and the documents and artefacts to be produced, some efforts toward adopting a more formal development process have already been made:

- the OO paradigm and technology has been adopted as the standard development technology. The company has migrated from a Clipper based environment to the Smalltalk programming language and environment.
- a general framework of working and utility classes has been built as the main development structure in order to ease code reuse. The classes framework has been acquired from an associated partner, responsible for its updating and maintenance.
- standards for class documentation have been defined and case tools to automatically generate documentation form source code have been adopted.
- a configuration management system supports the development process: at present source code and run-time environments have been put under configuration management.
- maintenance interventions are automatically reported in the release version document: this is enforced by automatic tools, in order to guarantee an up-to-date documentation.

Identified weaknesses are the following:

- Product requirements were collected in an informal way, expressing them in natural language as a result of several interviews with the (internal) customers.
- No final user feedback is required until a first prototype of the product has been developed.

With regards to the design phase, the FTC approach is mainly based on identifying which classes of the framework can be directly reused, and which classes have to be developed from scratch. Identified corrective actions are:

- Formalisation of the software development process, indicating phases and related outputs
- Formalisation of the development phases (requirement specification, analysis and design phase) by adopting a standard methodology and tools to support them

With the RESPECT project, FTC has chosen to focus on the requirement specification and analysis process improvement as it is perceived as a strategic step which can guarantee a higher quality of the products and higher customer satisfaction.

The plans and the expected outcome

The experiment is being performed over a baseline project named "PROUD" (*Progetto Reti per l'Organizzazione delle Unità Distributive* – Project Distribution Units Network Organisation). It involves the FTC Programmers team supported by people involved in management, auditing and marketing services.

The PROUD project is a pilot project, performed before transferring main results and the same experience to other market fields. In particular, agricultural co-operatives, social co-operatives, and production co-operatives will be involved in the next step, thus being the natural candidates for adoption of the same solutions, experimented during the PROUD project.

The project is nevertheless a real production project. A complete system to support the commercialisation, direct retailing and consumer loyalty is going to be developed. Two of the components of the entire system will be the "Fidelity card management software" and the "Sales data warehousing".

The completion of the PROUD project is planned to require about 800 person/days. Several companies are involved in the baseline project with different roles: two external companies, will develop some of the software components. Within the scope of the PROUD project, the FTC itself behaves as a client for the PICO project (*Progetti d'Innovazione tramite la Cooperazione* – Innovation Projects through the co-operation), a project focused on financial auditing.

From an operational standpoint, the RESPECT project is expected to have a twofold impact on the customer-supplier interaction. On the one hand it is expected to rationalise the definition of the needed software product characteristics, for the benefits of the software developers. On the other hand, since customers tend to provide imprecise descriptions of their real requirements, it is expected to facilitate customers in describing their final expectations earlier on in the prototyping phase.

Most importantly, from an organisational standpoint, the RESPECT project is expected to help overcome the resistance encountered during the introduction of the new methodologies, a problem which has been all too frequent in the past. In particular, past experience in adapting solutions to the FTC environment is being used to reduce the resistance to change and to minimise the overhead that the adoption of the new methodology introduces.

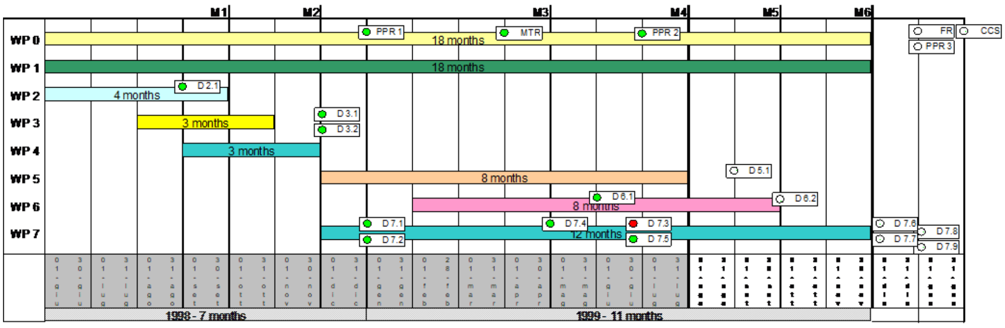
Finally, RESPECT is expected to enhance the software-related documentation. Currently the two selected components of the baseline project are the first ones in the several years FTC software development experience provided by requirements related formal documents. Until now, it is the very first time that informal verbose descriptions are substituted by a structured document, compliant with general requirement standards.

The implementation of the improvement actions

The experiment is being performed by defining a new requirement specification process that includes the use of automatic tools to be applied to the selected baseline project and by measuring the benefits obtained in terms of higher customer satisfaction, reduced effort per requirement, reduced time to market, reduced rework and higher software quality.

The basic approach is that after a training session in the new methodology and tools, an existing development team is employing the new techniques in the baseline project, comparing them against their own past experiences with a traditional methodology (i.e., comparing the situation with the existing development process).

As most EEC-funded projects, the work plan for RESPECT is divided into several work-packages. The most significant ones, from an implementation point of view are WP2 – Tools acquisition and Integration, WP4 – Training, WP5 – Experimentation, and WP6 – Analysis and Consolidation of results.



In Time Released	● D 2.1
Postponed	● D 3.1
To Be Done	○ PPR 1

Work Packages	
Reference	*Title*
WP 0	**Project Management**
WP 1	**Co-operation with other ESSI projects**
WP 2	**Tools Acquisition and Integration**
WP 3	**Predefinitions and Experimental Design**
WP 4	**Training**
WP 5	**Experimentation**
WP 6	**Analysis and Consolidation of Results**
WP 7	**Dissemination**

Deliverables		
Reference	*Title*	deadline
Del 5.1	**Requirement specification and analysis s**	**Aug-99**
Del 6.2	**Analysis report**	**Oct-99**
Del 7.5	**ITC-irst Presentation**	**Jun-99**
Del 7.6	**Third internal presentation transparencies**	**Dec-99**
Del 7.7	**Associated societies material**	**Dec-99**
Del 7.8	**Presentation to peers in Europe**	**Dec-99**
Del 7.9	**Final International presentation**	**Dec-99**
PPR 3	**Analysis and Consolidation of Results**	**Dec-99**
FR	**Final Report**	**Dec-99**

Fig. E.C. 1: The RESPECT Project Gannt

Although WP2 is not directly related to the organisational and business objectives, it is nevertheless an integral part of the implementation as it makes use of a methodology for tool selection (DESMET) that has been particularly important in leading the project staff towards a structured approach to requirements definition. The influence of adopting such a methodology is pervasive across the project – this is a positive side effect of its use. However, since it primarily relates to technical objectives, the results of WP2 are not presented in this article for brevity reasons.

The measured results and the lessons learned so far

At the time of writing, the analysis and consolidation of results (WP6) has not taken place yet and therefore only informal results can be presented. However, based on the feedback from the activities in WP4 and WP5, it is already clear that several results are being achieved in line with the expectations. The points worth noticing are presented below.

The operational stand point

The general guidelines for requirements definition are providing a structured reference for the requirements definition phase. In the experience of the involved staff, support to software development process is certainly enhanced.

The adoption of a couple of tools (Rational RequisitePro and IBM Visual Age) supporting the requirements definition phase and the analysis and design phase, has proven to provide an effective means for the development early phases. The availability of a company database containing previous projects requirements and designs is facilitating reuse, thus reinforcing the expectation for a shorter time to market for future applications.

The business standpoint

The new requirements specification phase is resulting in increasing confidence that the object model derived better implements what a customer/user really wants.

The new analysis phase supports a more natural partitioning of a complex system into smaller components, easier to be managed. This results in a better application definition in the design phase according to customer expectations, rather from later arbitrary design choices.

By improving clarity of the communication channel back to the user, the perceived quality of the product is improved by making an impact on customer satisfaction earlier. That is, the user can see from the clear representation of his expectations whether the system will satisfy the needs (rather than vaguely suspecting that they satisfy the need, but awaiting a completed prototype to verify the suspicion). In such cases, where multiple contracts may be under consideration simultaneously, the improved user satisfaction earlier in one contract may be the triggering event to facilitate the choice.

The total business volume for FTC’ software development team is increasing, thanks also to improved customers' confidence in the quality of requirements definition.

The organisational standpoint

As a result of the experience made within this PIE, some professional roles are being considered in order to better support the software development process. In particular, it will be probably defined a new responsibility about requirements definition and analysis phases.

More in general, a better defined and more structured development process is leading FTC to define roles and responsibilities more precisely, thus improving the structure of the organisation.

From another point of view, having identified a defined requirements responsibility, FTC is forced to carefully planning and scheduling of the ongoing projects, thus supporting a better defined development process.

The cultural standpoint

Staff generally agree that the new process is improving the final quality, and they also recognise that new methodologies are reducing the number of interactions between supplier and customer.

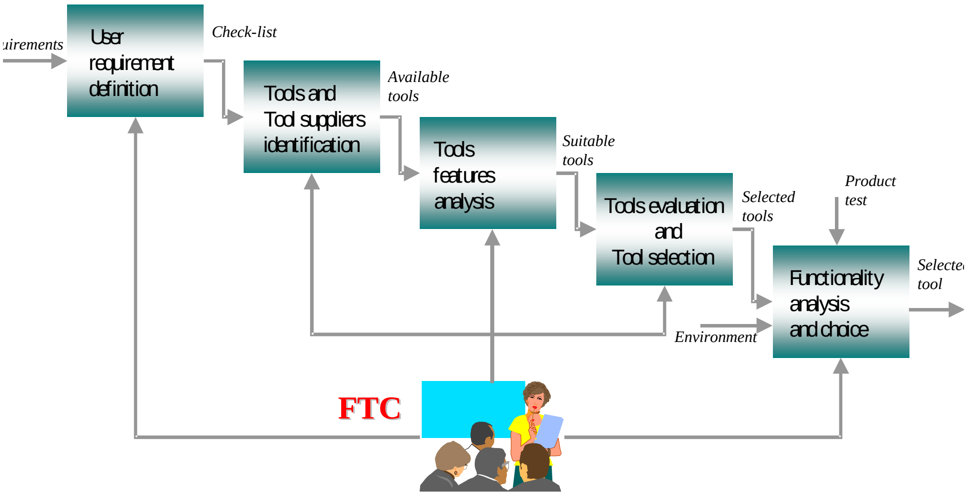
Staff also feel that the most part of these benefits will be probably available after that a number of projects will be completed, providing feedback and support for reuse. Until then, the introduction of the new methodologies risks to be perceived to increase the "project bureaucracy".

However FTC people feel that the project is perceived as the necessary first step to define a more formal development process.

The skills standpoint

People involved in RESPECT are gaining new skills and knowledge about several different fields and methodologies. For example FTC staff learned how to use the DESMET methodology for tool selection. After this experience, they applied the same methodology to screen and select other ones, choosing the best suited for a certain task.

Fig. E.C. 2: How DESMET is used by FTC


uirements
User requirement definition
Check-list
Tools and Tool suppliers identification
Available tools
Tools features analysis
Suitable tools
Tools evaluation and Tool selection
Selected tools
Product test
Functionality analysis and choice
Selecte tool
Environment
FTC

The impact is clearly evident at FTC and it is expected that the analysis to be conducted in WP6 will reinforce these findings. To conclude, however, a few considerations can be made regarding resistance to change.

All in all, FTC' experience is that methodologies need to be explained and adapted to be useful in practice. In particular, FTC developers have been trained about software engineering fundamentals. Only after the training sessions it has been possible to define general guidelines for the requirements definition phase. This is one of the main learned lessons. Training is one of the most important factors to overcome resistance to change.

Equally, the newly introduced tools have been used to define not only requirements and classes structure for a specific project, but for the whole of future FTC developments. A database of requirements has been established, that, although it is based on a few projects, already contains a number of requirements and use-cases defining typical interactions with the system. Another lesson is therefore that thinking ahead for reuse can really make a difference in the way the requirements are captured. In turn, facilitating reuse makes it easier to win developers acceptance of the new methods.

APPENDIX 1: Authors

Dr. Enrico Cozzio - Federazione Trentina delle Cooperative, Trento, Italy.
Managing Director of Ufficio della Cooperazione di Consumo Trentina, a department of Federazione Trentina delle Cooperative.
This department deals with accounting and auditing, and also offers financial and administrative help, together with organisation and planning strategies to retail co-operatives (Famiglia Cooperativa scrl) and their consortium (Sait scrl), in the province of Trento (Italy).

Dr. Francesco Calzolari - ITC - Irst, Povo (TN), Italy.
Francesco Calzolari received his Laurea degree cum laude in Computer Science from the University of Pisa, Italy, in 1991, with a thesis in the field of Fault Tolerance, and his Ph. D. in Informatics Engineering from the Politecnico of Milan, Italy, in 1996, with a thesis about Timed Petri Nets and Real Time Systems.
In 1997 he joined the Software Engineering group at the Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy. He was involved in the researches about dynamic models for software maintenance and testing.
Actually he is member of the STAR Project team (Maintenance of Software Systems) at IRST, and his current research interests include software engineering, dynamic models and reverse engineering.

APPENDIX 2: Companies

Federazione Trentina delle Cooperative (FTC)

The Trentino Region.

The Autonomous Province of Trento is situated in northern Italy, in the Alps, with a population of 464,000 inhabitants over a mostly mountainous area of 6,207 km sq/.

Out of 223 council towns, only 5 have over 10,000 inhabitants, and they assimilate 38.9% of the population.

Trento, the capital of the region has over 100,000 inhabitants and the town of Rovereto 33,000 inhabitants, both of which are situated in the valley of the river Adige.

The Co-operative Movement in Trentino.

The Cooperative Movement in Trentino was born at the end of the last century, based upon the economic theories of Friedrich Wilhelm Raiffeisen.

The Trentino area was a fertile breeding ground for the development of these ideas, whether due to the culture of the population or to the favourable administration, ever since its recognised origins, (in that which was a province of the old Austro-Hungarian Empire), thus so today under the administrative bodies of the Autonomous Province of Trento and the Region Trentino Alto Adige/Süd Tirol.

The Coop. Movement, within Trentino culture, is the economic consequence of the spirit of associationism, of voluntary work, of good will and the capacity to “do things together”, in essence, of the culture of self -government and self-management deeply rooted in this Land and tangible in everyday life.

Therefore, Trentino is a land of consolidated co-operation, so much so that out of 464,000 inhabitants more than 160,000 are co-operative associates (100,000 not counting double or triple subscribers) which traditionally operate in the following sectors.

- **Credit.** With a network of over 330 counters, the 80 Casse Rurali (Savings Banks)are present in every council towns of the province and they assimilate globally 60% of the market. The activity of the system is coordinated by the **Cassa Centrale** (Central Bank), a consortium of all the Casse Rurali of the Province, which acts as a centre of service and a common meeting point.
- **Retail Co-operatives.** With a sales network of 324 shops the 120 Famiglie Cooperative (Family Cooperative Stores) operate in every council towns of the province and assimilate globally 35% of the market. The activity of the system is coordinated by **SAIT**, a consortium between all the Famiglia Cooperativa of the Province.
- **Agriculture.** By means of the Cantine Sociali (Wine Producers), the Caseifici Sociali (dairies), the Magazzini Frutta (Fruit Warehouses), the cooperative collects, handles, conserves and distributes over 80% of the agricultural produce of the entire province. The individual co-operatives are then guided and coordinated in the running of their activities by sectorial consortiums: Apot (divided into Melinda, La Trentina, Piccoli Frutti) for fruit, CAVIT for viticulture, and CONCAST for dairy production.
- **Production, work, services.** An emerging Sector which consists of C.T.A. (Land and environment Association), the reference point modelled upon the traditional sectors cited above.
- **Social Services.** These are the co-operatives of the future and supplement the deficiency of the welfare state in caring for the needs of assistance, even for individual

cases, of marginalized subjects, helpers for the handicapped or the sick. Even this sector has its own reference point Consortium, Consolida.

The Trentino Federation of Co-operatives.

The Trentino Co-operative Movement is assisted, co-ordinated, guided and controlled by the Federazione Trentina delle Cooperative (Trentino Federation of Co-operatives), a consortium amongst all the co-operatives of the province and their consortiums.

It is precisely the guidance and co-ordination of the Federation which confers upon the Movement of Co-operatives the combining power of the system and the claims of the economic model, according to the ideology which inspired Raiffeisen.

The Co-operative Movement in Trentino employs 7,788 permanent staff and 2,752 for seasonal work.

ITC - IRST

ITC-Irst, the RESPECT subcontractor, is a public research institute whose activities include software engineering and maintenance. ITC - Irst holds a solid background in software engineering issues, especially in object-Oriented modelling, effort estimation, static and dynamic code analysis as well as in software metrics. Several tens of articles presented at international conferences or published by scientific journals the impact of such activities on the scientific community.

Within the RESPECT project, ITC - Irst co-operates with the FTC for scientific and methodological aspects, supporting activities regarding tool selection, customisation and training, implementation as well as requirement process and guidelines definition.

Software process improvement using CASE: lessons from the front-line

David Wastell
University of Manchester, UK
Chris Williams & Mike Willetts
Salford City Council
Karlheinz Kautz
Norwegian Computing Centre, Oslo, Norway.
Peter Kawalek
Warwick Business School, UK.
Tom McMaster
University of Salford, UK.

CAPELLA: context and aims

CAPELLA (CAse tools for Process Enhancement in LocaL Authorities) was initiated in March 1997 as a result of a successful bid the previous autumn to carry out a process improvement experiment (PIE) under the European Union's ESSI programme (European Systems and Software Initiative). The overall aim of the project was to develop a new methodological framework in Salford City Council's IT Services Department (ITSD) for developing software based on the use of a CASE tool (Oracle's Designer 2000, D2K). It was expected that the project would lead to an in-depth appreciation of the implications of implementing a CASE tool and associated methods, and engender a wider understanding of the issues and impacts of technological change within an organisation. An interesting feature of CAPELLA is that it represents a collaboration between the Council and academic researchers from local institutions, and also the Norwegian Computing Centre.

The rationale of this paper is to narrate a frank and honest history of the project and to highlight its achievements in terms of lessons learned and its lasting legacy.
At the outset of the project, the software development group in ITSD consisted of around 28 software professionals organized in three teams under a single manager. They were responsible for the development and maintenance of software systems for user departments throughout the Council, the principal users being the Directorates of Education, Housing, Social Services, and the Treasury. Although the department had a generally good reputation amongst its user community, there were pockets of dissatisfaction and ITSD had a mixed reputation for performance. The quality of software produced was generally acceptable. However, timeliness and budgeting targets were regularly exceeded and the customer departments felt that ITSD failed to provide sufficient information with regard to progress and project budgets. Interestingly, where strict deadlines were imposed, for example through legislation, those projects were delivered on time.

It was hoped that the use of CASE within a systematic methodology would achieve tangible benefits in terms of both improved developer productivity and enhanced software quality (especially in terms of user satisfaction). Productivity would be enhanced by two primary means: through the standardisation of working methods and through the technical facilities provided by CASE, e.g. integrated analysis and design tools, automatic code generation, a central code repository enabling more re-use. Improvements in software quality were sought in terms of both its technical dimensions (maintainability, cost of ownership etc.) and in terms of a better fit with user needs. It was hoped that CASE would directly improve technical quality and would indirectly support better business alignment by enabling higher levels of user participation in the development process (primarily to be achieved through the ability to prototype rapidly). The project aspired to introduce a new software paradigm, described as the “Total Team” approach. It was hoped that the above benefits would also translate into increased developer job satisfaction resulting in staff retention and improved customer satisfaction resulting in repeat business.

In a nutshell, the original CAPELLA plan was structured in two phases. It was recognised that the introduction of CASE and a new methodology represented a major change. Driving the implementation strategy was a concern to develop internal capability, not to become dependent on outside experts or consultants. This approach would also maximise feelings of ownership and minimise internal resistance. Accordingly, the first phase of the project (initial 6 months) aimed at creating such internal capability in the form of a Centre of Excellence (CoE). Key skills would be developed by using CASE on a pilot project. Once the skills were in place, CoE personnel would then play the role of internal consultants assisting in the deployment of CASE (and new working methods) throughout the rest of the department (months 6 to 18).

An unexpurgated account of the project in terms of its main events now follows. The account is broken into 6 month segments (i.e. semesters). Dates are approximate.

Semester 1: First steps (inception to late summer 97)

The early phase of the project was problematic. Following a well-publicised and relatively well-attended launch event (by both ITSD staff and to a lesser extent users), the project went into a quiescent phase. During this time, the project was being managed by the Head of Software Development (HSD), who had played a supporting role in the original bid. Although the CASE tool had been procured and some staff trained in its use, work on the baseline project, a major integrated system for the Housing department, stopped due to rapidly escalating costs and timescales, and the customer department took the decision to review the market to compare new systems with the in-house development. Ultimately, they decided to purchase a package solution.

Although an alternative project was identified concerned with the financial management of regeneration projects (to be known as PROJ_1), both momentum and enthusiasm had been lost. The deployment of CASE on a major project was critical to the whole strategy of CAPELLA. It would mean that many developers would ultimately be involved in its use; seeing CASE deployed on a flagship project was also of obvious symbolic value in showing how central CASE (and CAPELLA) was to ITSD's long-term strategy. Use by a small group on a small project suggested the opposite. However, this should not be seen as a tactical error so much as a genuine change in the business environment. The loss of the Housing project was simply a reflection of a more general trend affecting ITSD (and internal IT/IS departments in general), namely a move away from in-house bespoke development to outsourcing and the use of packages. That CASE was coming to be seen as marginal was thus accurately reflecting the wider business realities.

Summarising this period, the main positive outcome was that a small group of staff had been trained in the use of D2K, and that the CoE now existed in the form of this group. Little work on methodology had been initiated, although some preparatory work on metrics had been carried out (for evaluating quality and productivity gains). The main user departments had been interviewed and an internal investigation of the use of function point analysis (Fenton, 1991) had been undertaken (with generally positive findings). Although a CoE existed, it is fair to say that it (and CAPELLA in general) lacked a high profile within the department as a whole, with most developers seeing it as having tangential relevance to their work. The primary reasons included: the loss of the Housing project and the general lack of appropriate development projects; lack of strong project leadership; low morale amongst developers leading to a general weariness in respect of any innovation.

Towards the end of this semester, HSD left his position at Salford and there was a major re-organisation within ITSD. The Customer Services and Software Development business groups were merged with the Customer Services manager (HDCS) taking on the head-ship of the newly formed "Development and Customer Services" business group. HDCS had led the original bid on the Council's side. Availability of staff resources was obviously affected during this re-organisation, and a key member of staff also left the CoE for a position in the private sector.

Semester 2: The Salford Methodology: a false dawn? (autumn 97 to spring 98)

The second semester of the project was characterised by a switch in focus towards the development of the methodological framework that had figured as the second main element in the original proposal. CASE development work continued on the baseline project and a further project was identified (PROJ_2); however, no significant expansion in its use occurred. Although training requirements for CASE were examined, little actual training occurred over this period either internally or externally. A member of ITSD staff was designated to work full time on the methodology (PR1). Prior to Christmas, the main event was the holding of a workshop, attended by several development staff including PR1 and one team leader, on Soft Systems Methodology (Checkland, 1981); this was seen (by the academic researchers) as a tool that could form an important element of the new framework.

Following Christmas, a concerted attempt was made to define the new methodology, drawing on best practice in past projects. A structured methodology focused on the use of CASE, with 14 stages covering the whole life cycle, was mapped out as a joint exercise by a team comprising one of the researchers, a senior practitioner (PR2) and PR1. Two of the stages were defined in detail. This work was however curtailed following a visit by PR1 and another member of the CoE (PR3) to a D2K workshop where they were introduced to CDM, Oracle’s proprietary methodology associated with D2K. This appeared to hold much promise as a potential framework. It was defined in detail, appeared to conform well to the kinds of development projects handled by ITSD, and was geared to the application of D2K. “Why invent a methodology of our own” was a persuasive argument which re-directed effort to the evaluation of CDM. The decision was made to apply it to PROJ_2 and retrospectively to another project, the ultimate aim being to customise it to ITSD’s working practices. PR1 was to take the lead in this.

This work continued over the latter part of this semester. Some progress was made but ultimately the decision was made to discontinue this line of work. PR1 showed increasing disillusionment with CDM and with CAPELLA in general. CDM appeared to consist of no more than a set of Word templates for holding documentation; in detail, it did not conform well to the standards in place within ITSD. It was also unclear as to whether there was any real need for CDM, given both the dearth of development projects and the marginal position of CASE. The sense of an impending crisis was becoming strong and in late spring a series of meetings was held which were to betoken a major change in the direction of the project.

Semester 3: The renaissance of CAPELLA (spring 98 to late summer 98)

The conclusions of these meetings were twofold. First, that lack of internal resources had impeded project progress and that a significant injection of new effort was required to drive forward the technical work of the project. However, experienced staff were not available given the extreme demands on the department at that time arising from the mainstream of its work: the maintenance and upgrading of legacy systems, package development, and increasingly, Y2K auditing. The limited role foreseen for CASE made it difficult to argue for more internal resource at such a time, to progress its further deployment or the work on methodology. The second conclusion was to switch the focus of the work onto activities that did appear to be of real benefit to the department and to reappraise the work that had been done, whilst remaining consistent with the general thrust of the project.

Early summer saw a major restructuring of the project. A senior member of ITSD was appointed to lead the project. Two students were recruited from Manchester University to carry forward the bulk of the technical work, and a further researcher was engaged. Three main lines of work were mapped out: a codification of the methodological work that had been done, the development of a set of software metrics using the GQM approach (Basili, 1995), and an assessment of the lessons learned from the experience of implementing CASE. ESSI were informed of the status of the project and a request was submitted for a 6 month extension on the basis of past problems, the proposed restructuring and the future benefits. This was granted.

Work proceeded on metrics and methodology over the summer, and resulted in several reports, which formed the inputs for a 2 day workshop in early October at which the objectives and work programme for the remainder of the project were defined. Two reasons for the desultory progress made in the project were identified. The decline in in-house development was re-affirmed as a major inhibitory factor; this had rendered both CASE and the work on a development methodology of increasingly marginal business significance. A second factor also received extensive discussion, namely the low level of software engineering discipline that appeared to prevail in ITS. This had been highlighted as a result of a CMM assessment (Humphrey, 1995) performed over the summer which showed ITSD to be at level 1, i.e. chaotic. More tellingly, the lack of formality had come to light as a result of an attempt to introduce some basic mechanisms for project control; an analysis of current commitments had revealed that much of the department's work was unofficial (up to 50%!) in the sense that no record was present in the formal order book.

Given these factors, the conclusion of the workshop was to focus the remainder of the project on process improvement areas that were desirable and attainable given the working practices and culture that prevailed in the department. Work on a monolithic development methodology was no longer regarded as appropriate and it was resolved to re-focus the methodological work on key practices. This was felt to represent a more flexible approach allowing the department to improve its performance in discrete areas as part of an on-going process of continuous improvement attuned to the exigencies of its business environment. Each year, one or more practices would be targeted for focused effort.

Two such practices were identified for the remainder of CAPELLA. The first, and the most important in terms of effort, was a metrication initiative aimed at putting in place a simple set of metrics that would provide a rudimentary degree of project monitoring and customer feedback. Part of the inspiration for this was to enable the efficacy of CASE to be evaluated (to the limited degree that it had been deployed). The second motivation was that this would be a significant move towards the creation of an effective project management infrastructure. Although some project control mechanisms were in place, we have seen that they needed strengthening and formalising. In practice many projects were initiated without plans or indeed formal approval, and where plans existed there were wide variations in the degree to which projects were monitored or controlled against those plans.

The second key practice was peer review. Two benchmarking studies recently carried out by one of the researchers had identified this as a practice which had been successfully adopted by two other level 1 organizations, and which had led to real improvements in software quality.

At this point, a further significant development will be mentioned which had occurred in the early part of the semester, namely the inauguration of a Standards and Methods Group (SMG) within ITSD. The remit of SMG was to identify existing standards and methods, to develop new ones where appropriate, and to promulgate their use. Membership of SMG was open to any member of ITSD; participation in its work was voluntary. CAPELLA and

SMG were clearly related in the sense that both initiatives were aimed at improving software practices and a formal link was established. It was resolved that CAPELLA activities would be progressed in concert with this group, and that we would draw on any relevant standards work that was being done by SMG.

A presentation of the results of the workshop was made to SMG. Several staff members volunteered to work on both the metrics and the peer review strands. The two students were retained as ITSD staff (RA1 and RA2); their primary remit was to carry through the metrics thrust of the project. One of the academic researchers took lead responsibility for progressing the peer review initiative.

Semester 4: The denouement (autumn 98 – spring 99)

The October workshop had generated some provisional proposals regarding metrics. Two classes of metrics had been distinguished: developer metrics and customer metrics. The former focused on project deliverables and key products, measuring such features as the planned/actual effort to produce deliverables, planned/actual delivery dates, software complexity, tools used etc. Customer metrics represented the users' evaluation of a software system in terms of its ease of use, its reliability, maintenance cost, and so on. Work in these two areas was progressed separately, and will be reported in the following two sub-sections. The results of the peer review work will then be summarised.

Developer metrics

6 projects were selected on which to pilot the developer metrics. 3 of these were CASE projects, including the two projects already mentioned, namely PROJ_1 and PROJ_2. The third was a new CASE development. A non-CASE project was also selected for comparison, and two package development projects were included on the same basis. This pilot experiment began in mid- November, the aim being to run for 3 months and then to analyse the data and evaluate the effectiveness of the initiative.

Major problems were encountered immediately in that only one of the 6 projects was actually live and running, namely PROJ_2. The new CASE project had been cancelled, the two package projects were in abeyance, and the other two projects were finished. Collecting the proposed set of metrics on PROJ_2 also turned out to be infeasible, for a variety of reasons. Some metrics were seen as "too complicated" (i.e. it appeared impossible to devise a standard method for assessing the complexity of design products such as DFDs given the wide variations in way that they were produced); other metrics were abandoned on the grounds that no relevant and reliable information could be found in the existing project documentation (e.g. type of deliverable, delivery dates etc.).

In the light of these problems, the original set of developer metrics was scaled back to a very limited subset, effectively providing timesheet information recording how much effort was expended on different activities (e.g. logical design, data conversion) for a given project. A detailed booklet was issued to the PROJ_2 team members (3 in number) indicating how the timesheets were to be used, and the data collection exercise was then initiated. The trial itself was disappointing in that very little work on PROJ_2 occurred over the course of the experiment, and the team members showed decreasing willingness to complete the sheets. Only 5 timesheets were ever returned. However, the experiment was invaluable methodologically in that it stimulated the development of a high level lifecycle model for CASE development, in which the development process was divided into 15 tasks (e.g. produce logical design) grouped in 8 stages. By the end of the pilot, this Process Model had been through a series of refinements and was seen to conform well to the work the developers did.

Overall, the trial was a useful learning experience, which had demonstrated the general feasibility of collecting timesheet data in real time, providing that an accurate process model of the type of project was defined. The decision was made to develop the approach further and to implement it across the whole of ITSD for a one month trial period. The RAs devised high level process models for each of the 10 types of project undertaken by the department (CASE development, non-CASE, package work, Y2K testing, database upgrades etc.) drawing on such standards that existed (e.g. a standard for package development had been developed by SMG).

In the end, this experiment ran for 8 weeks. Data was successfully collected for a high proportion of ITSD personnel, peaking at 20 individuals over the first 4 weeks before tailing off to 10 for the final 4 weeks. Exploratory analyses have shown that usable information was gleaned (e.g. accurate pie charts showing the distribution of time and effort across the different types of project work). The validity of the various process models has also been examined by checking if a valid task type has been specified against each timesheet entry. For many processes, validity was high: for CASE development, for instance, 85% of entries were valid, for non-CASE, 87%. The figures for other major processes such as package development (59%) and Y2K work (65%) were somewhat lower, indicating that further refinement of the process models and standardization of work practices is required in these areas. The RAs also investigated a number of other issues of interest, such as variations in task length across teams.

Customer metrics

The methodology for the definition of the customer metrics is of interest, the aim being to produce a questionnaire instrument for assessing customer satisfaction in relation to their use of IT systems. Following a review of seminal papers from the research literature (e.g. Doll and Torzadeh, 1988) a prototype questionnaire was constructed. This was shown to a group of volunteer users who provided valuable but relatively minor feedback, e.g. that clearer topic definitions needed to be provided and a more logical grouping of questions was required. Subsequently a focus group meeting of user representatives was held which helped to clarify the concept of quality from a customer perspective and provided further feedback on the service provided by ITSD.

The resulting questionnaire contained a number of sections, grouped in two main areas: past experience of software development (users were asked to rate how involved had they been in the development of the system they used, how much control they had had, how responsive were ITSD etc.) and their levels of satisfaction with the resultant system (the support given, accuracy of the data, informational content, format, ease of use and flexibility). The motivation for the first set of question stemmed from our conviction that user participation was the key to successful software development (the Total Team approach) and that CASE was a means to achieve this.

Customer data was then collected by distributing the questionnaire to the users of a sample of extant systems. It had been hoped to carry out a comparison of CASE vs. non CASE systems; however, no CASE systems were operational at the time of the survey. Hence it was targeted on non-CASE systems only, 3 in all operated by two departments (Corporate Services and Social Services). Nonetheless, the exercise was seen to be of value in that it would provide a baseline for the future and that it would provide feedback on feasibility and validity of the methodology.

Questionnaires were distributed to the identified contact person for each of the systems; they were asked to pass them on to staff who were users of the system and/or had been involved in its development. Questionnaires were returned from only one of the departments; 27 responses were received from an estimated 70 distributed. Regarding the first section of the questionnaire, respondents were overwhelmingly positive about their experience: e.g. 75% of responses indicated an appropriate level of involvement and control, 85% indicated that user/developer relations and communication had been good. However, the number of respondents here was relatively small. The systems themselves were judged to be satisfactory in most respects: support (86% positive responses), accuracy against specification (100% positive), content (99%), output format (92%). For ease of use (67%) and flexibility (50%) the proportion of positive evaluations was somewhat lower.

Whilst these results are generally positive, and indicate a highly suggestive relationship between participation in development and project success, they reflect attitudes to only two systems in one department. They broadly confirm the findings of our early interview work that this department believed strongly in a user-led approach, that they committed resources appropriately, and that they were very satisfied with the systems that they currently use.

Peer review

Peer review can be defined as a structured, quality improvement process whereby a team member submits an item of work for constructive evaluation by a group of colleagues, who may be team members or indeed drawn from outside the team. It was seen as bringing several major benefits to ITSD: as a tool to improve quality, as a way of disseminating good practice (including new standards), a mechanism for strengthening teams, and a means of sharing critical knowledge so that it was no longer “locked-up” in the heads of single individuals.

Several members of the SMG worked alongside one of the researchers in an effort to define a standard procedure for peer review and to evaluated its usefulness. Two experimental reviews were carried out, which were seen to be valuable, and a standard was defined which outlined a methodology for peer review (preparatory work, who should attend, key roles, the review format, guidance for reviewers, reporting and feedback). Quality criteria for a set of high priority topics for peer review were also outlined, including feasibility studies, project definition documents, and design documents.

The legacy of CAPELLA

In this section we will take stock of the results of CAPELLA. For each major area of work, we will comment on what was ultimately achieved and what is planned for the future. We will conclude with some general reflections on what has been learned in relation to the management of technological change.

A resume of technical achievements

Although the original grand design for CASE has not been fully realised (i.e. to provide a common platform for all application development) nonetheless a CASE tool has been acquired and expertise developed in its usage. Although this expertise is confined to a small group of staff and has been applied on a minority of projects, it is likely that the demand for these skills will continue, and should larger scale in-house development projects materialise, a core of experience has been established. Informal interviews with developers have revealed positive attitudes towards CASE and it is felt to have enhanced job satisfaction; on the technical side, its ability to produce high quality documentation is felt to be a particular benefit. Quantitative assessment of the benefits of CASE has not been possible for reasons given above; i.e. that no CASE developed systems are presently in use.

The Total Team approach reflected our philosophical commitment that user participation is the key to achieving high quality software systems, and the interview/survey data confirmed the importance of a user led approach. Our original plan had been to develop a semi-structured development methodology exploiting the features of CASE (e.g. prototyping) to promote more intensive user involvement. In practice, achievements in this area have been limited, given the changing priorities over the course of the work. A methodology was defined, but only at a high level. The move away from a monolithic life-cycle methodology towards key practices reflected the need for a flexible, modular approach to improving working methods. Establishing the key practice approach is an important result; it provides ITSD with a general framework for continuous process improvement that will stand them in good stead in the face of a highly dynamic and increasingly customer-oriented business environment. The modularity of the framework is important as it enables incremental changes to be made that are attuned to the availability of resources and the prevailing business priorities.

The Peer Review initiative exemplifies the Key Practice approach. The introduction of this practice was felt to be an important step towards improving software quality, and the results of the experiment were promising. Peer review was seen to be a useful tool and a draft standard is now in place, including document templates. There is a clear intention to proceed with its implementation, following further refinement of the methodology. Questions being debated include: what products to apply it to, when should it be done (mid or end of phase), how should it be followed up? As a first step, it has been proposed that all projects must include at least one peer review built into the project plan.

Both metrics initiatives led to positive results. The timesheet experiment was a success in that data was collected over an short but not insignificant period, and that the information generated appears to be both valid and of intrinsic interest. However, the experiment also indicated that staff compliance will be problematic and that more detailed consideration needs to be given to the content of the information to be collected from both management and operational perspectives. The experiment thus demonstrated the feasibility of the methodology; it provided valuable experience and generated a set of useful document templates and process descriptions. A system for recording effort against projects in real-time is important for resource management and it is likely that, after further refinement, the timesheet mechanism will form an important component of ITSD’s embryonic project management system (see below). Timesheets were also felt to be beneficial as a "self-productivity" check.

The results of the customer metrics study were also positive in that the metrics appear to be valid and meaningful, showing that customer satisfaction is measurable not only in relation to the systems developed but that the development process is capable of metrication too. Valuable feedback on the methodology was obtained as a result of the experiment, indicating it to be generally sound, although there are problems that remain to be addressed: e.g. the absence of responses from one department was a major flaw that needs further investigation, with the methodology to be improved as a result. As a result of the experiment there is a clear intention to refine and implement the customer metrics as a routine feedback mechanism, and a further survey is planned for next year.

The October workshop identified an urgent need for improved mechanisms for project management and CAPELLA has certainly instigated important work in this area. Apart from the timesheet experiment, work on the creation of a standard for project management has commenced over the last semester, under the aegis of CAPELLA. Three concrete achievements may be noted:

- a detailed set of documents has been written defining the draft outline of a project management system (including a detailed set of pro formas for recording information);
- it is a requirement now that all projects have a Project Definition Document, setting out scope, objectives, risks, resources, timescales and milestones (the PDD is seen as an obvious priority target for peer review);
- it is also required that all projects produce monthly progress reports, using a form modelled on the report used by ESSI to monitor periodic progress (PPR).

These latter two documents (the PPR especially) represent real concrete achievements in the sense that they have continued in use after the end of CAPELLA, and although still uneven there is a steadily rising trend in the quality of the reports being produced.

As a final achievement, the inauguration of the SMG should be noted. CAPELLA in part inspired this, in the sense that SMG's creation reflected an awareness of the need for self-reflection and process improvement that CAPELLA has helped to engender. In terms of its work, the liaison with CAPELLA has been of reciprocal benefit; standards generated by SMG have been used within CAPELLA and the results of our work have been reported back to SMG. In this sense, CAPELLA has strengthened the role of SMG. It is felt that SMG has proved itself as an effective forum, although it has declined in vigour over the last six months owing to the voluntary basis on which its technical work was done; in the future, SMG projects will be treated on the same basis as any other project and resourced fully and formally.

Final reflections on the management of change

The foremost observation to be made about CAPELLA is that it was a highly ambitious project, the aim being to make radical and comprehensive changes to working practices in a sizeable IT department over a short period of time. That the outcomes can seem modest in relation to our original ambitions reflects more on the magnitude of our goals rather than the lack of real achievement. The project has undoubtedly been a valuable exercise which has raised awareness in many areas, generated important pockets of new expertise and has produced tangible outputs, some of which have already been incorporated into practice and others which are likely to have a significant impact in the short term.

The first part of this report presented a detailed history of the project. The reason for recounting such a blow-by-blow narrative (rather than the usual bland and abstract account) is to provide a body of data to reflect upon regarding the lessons to be learned from the project for the introduction and management of technological change.

Arguably the most important lesson is the importance of strong alignment of new technology with the short-term demands on the business and its long-term goals. Although both are important, our experience shows that immediate exigencies tend to take precedence over future aspirations in terms of their claim on attention and resources. However important CASE and the Total Team approach might have been in the long term, the limited nature of our achievements in these areas in large part reflects the lack of relevance of CASE and the methodology to the prevailing demands on ITSD. Our experience has emphasised the high degree of dynamism in today's business environment and that vigilance and constant effort are required if change initiatives (such as PIEs) are to remain congruent with changes in the internal and external environment. We have seen that gaps can easily open up, even over relatively short time-scales, and that delays in addressing these can seriously jeopardise the improvement initiative.

Our results underline the truism that complex technologies require a greater investment of implementation effort in terms of training, changing working practices etc than simple innovations. The benefits at an organisational and individual level will take longer to realise and the short-term costs will be greater and more inhibiting. Not only is more effort required but the learning and dislocation entailed in the immediate term are likely to engender decrements rather than improvements in performance. This is a serious problem for complex technologies such as CASE; the business risks of deploying these innovations on business-critical projects is very inhibiting. The lack of adoption of CASE is only partly explicable by the paucity of development projects. It is arguable that it could have been more widely used than it was, and we may attribute this to the investment of effort that would have been required, in terms of training and in the standardization of work practices in an organisation characterised by highly informal methods. These concerns have almost certainly held back its wider adoption, and the question is a moot one of the degree to which it would have been deployed on the original baseline, given the risks involved. These considerations indicate that incremental approaches to technological change are more likely to succeed for complex innovations. This was reflected in our original plans to roll-out CASE on a team by team basis. It is also reflected in the positive reaction to the Key Practices framework.

Leadership in terms of both vision and implementation has also been shown to be crucial to success. Arguably, leadership was lacking in the first semester of the project and this was a cause of some of the problems encountered. Implementation is critical, and our experience has shown how problematic this can be and has given important insights into the critical success factors. Ownership is at the heart of this issue; innovations will be adopted more readily if people personally identify with them. This lay behind our original decision to make use of internal personnel. Although external resources can give momentum to experimental initiatives (as we saw in the last semester) there is a danger that the these efforts can be come marginalised because they are executed by outsiders. It is significant that, although there are plans to implement metrics, the only initiatives from the last phase that are currently institutionalised are the PDD and the PRR: the implementation of both these documents was championed and led internally.

Let us examine further the success of these two innovations in terms of our previous analysis as the contrast with CASE is revealing. In terms of relevance, the benefits are clear: ITSD will find it more and more difficult to survive in an environment increasingly dominated by commercial imperatives without stronger control systems. The PDD and PRR represent important components of a nascent project management system that is crucial to survival. Relatively simple technology is being used (simple paper documents) and, as we have said, leadership is internal and at a senior level. The attention given to implementation has also been crucial and shows the importance of monitoring and accountability. Articulating a vision for change is not enough; implementation must be followed up rigorously through management structures with staff held accountable, especially middle management. PDDs and PRRs are now mandatory requirements in ITSD and team leaders are held accountable for their production. Despite early problems of incomplete recording, there is a steadily improving trend in the quality and timeliness of these documents.

References

[1] Basili, V., Applying the Goal/Question/Metric paradigm in the experience factory. In: Fenton, N. et al., Software *Quality Assurance and Measurement,* International Thompson Computer Press, 1995.

[2] Checkland, P., *Systems thinking: systems practice*. Wiley, Chichester, 1981.

[3] Doll, W. and Torzadeh, G., The measurement of end-user computing satisfaction, *MIS Quarterly,* 259-274, 1988.

[4] Fenton, N., *Software metrics: a rigorous approach*. Chapman-Hall, 1991.

[5] Humphrey, W.S. *A Discipline for software engineering*. Addison-Wesley, Reading, MA, 1995.

Session 6
SPI and Virtual Team and QA Systems

Chairman
Timo Mäkinen

Pori School of Technology, Pori, Finland

Experience with Process Improvement Collaborations in Distributed Virtual Work Environments

Klaus D. Zesar, Hyperwave Software R&D GesmbH., Austria
Kzesar@hyperwave.com
Dr Richard Messnarz, Director ISCN Ltd., Dublin, Ireland
Rmess@iscn.ie, rmess@iscn.at
Gabor Nadasi, Rainer Bernhard, NQA Developers, ISCN, Graz, Austria
Rbernhard@iscn.at, gnadasi@iscn.at

Abstract:
This paper describes (a) research work about virtual organisations performed at the University of Technology Graz, and (b) experience with an actual implementation of a virtual organisation for quality assurance for the company Hyperwave.

It contains a description of the underlying principles of a virtual organisation, the paradigms followed in a system called NQA (Network based Quality Assurance) to establish such a virtual collaboration for the purpose of shared quality assurance, and presents an outlook into the future of virtual organisations.

Richard Messnarz, Dr., Florence House, 1 Florence Villas, Bray, Co. Wicklow, Ireland, ph.: +353 1 286 1583, fax.: +353 1 286 5078

Klaus D. Zesar, Hyperwave Software R&D GesmbH., A-8010 Graz, Austria, ph.: +43 316 820918, fax.: +43 316 820918 99

Introduction into Virtual Organisations

What is a Virtual Organisation

Nearly most methods for improving and managing software processes have been focused on single individual organisations and their processes. Little attention has been paid to software development projects which involve a number of organisations of varying size and software process maturity. Companies which form strategic partnerships in situation- and target-depending ways, are commonly referred as ad-hoc-organisations or „virtual organisations“ [1]. Such temporary forms of partnership are more advanced, because an ongoing change of partners requires more flexibility in terms of company culture, communication and information management [19].

Organisations that join their core competencies together to carry out a specific software project build a software development co-operation network. Each core competence is represented through a special software process called core-process performed by its organisation. A core process is organised as a service within a company and is available throughout the whole virtual organisation. Others can order these services and do not have to worry about the availability of certain persons at certain times. The virtual organisation itself uses a pool of such services. The best processes for each part of the project are connected to a co-operation network by the initiator of the virtual software organisation. For a virtual organisation the effective configuration of these processes is of essential interest to be able to produce products in an efficient way, for which customers are willing to pay for [5].

Given that it is not likely a virtual organisation will form by itself, a moderator or broker is needed. The moderator initiates businesses, discovers technological gaps in the involved companies, finds a common language between suppliers and customers, and assists the initiator in an organisational and technological way to establish the virtual organisation [20].

Before a project starts, the organisations involved have to sign a general agreement which contains the requirements and goals. This contract is created out of a common understanding of business and in relation to the confidence of one process in the performance of the other. This understanding includes the product specification, the terms of co-operation, and the specification of social behaviour [1][15]. If there is a common understanding of business and the necessary portion of trust between co-operating partners written contracts are not necessary [6]. After agreeing to the contract, the supplier takes full responsibility for the fulfilment of the required performance, and for the performance of its suppliers to achieve maximum customer satisfaction as the primary quality goal.

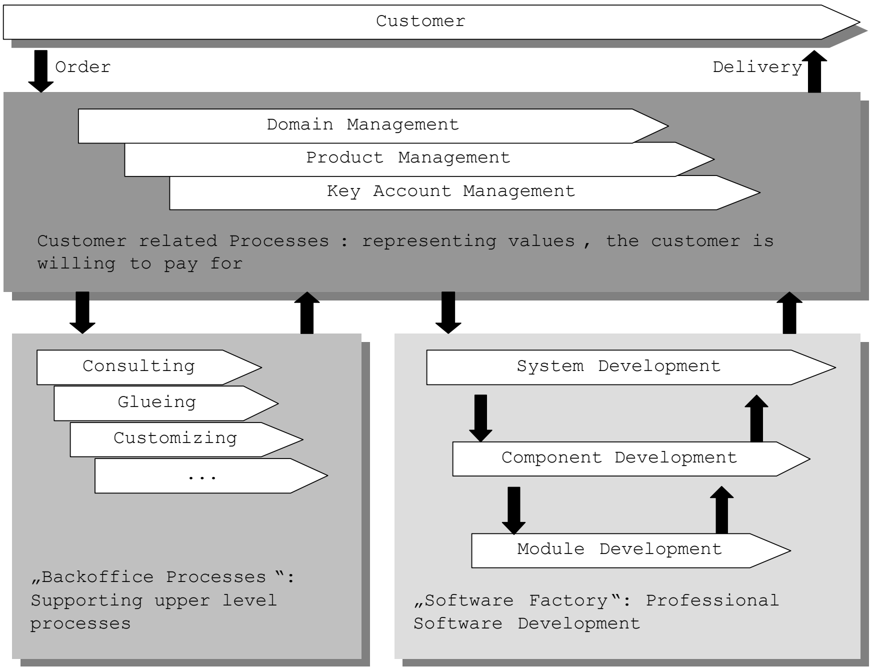

Customer
Order
Delivery
Domain Management
Product Management
Key Account Management
Customer related Processes : representing values , the customer is willing to pay for
Consulting
Glueing
Customizing
...
„Backoffice Processes ": Supporting upper level processes
System Development
Component Development
Module Development
„Software Factory": Professional Software Development

Figure 1: Generic sample of a process oriented software organization [24]

How does a virtual software organisation look like? Each virtual software organisation generally consists of three main components (see Fig.1) with different goals and tasks that are used to organise the software project [25].

Customer-oriented processes define an interface between the customers and the software development process. Their rate is to support the customer, analyse the customer's requirements, manage the services, projects and the product.

Software development processes are organised in a so-called software factory, much like a factory for mass production. Inside the software factory, software products are produced with the help of standardised computer-based tools. They use formalised processes that are controlled by technical and economical metrics [2]. These processes are optimised for a specific area of making them in appropriate for other application areas.

Additional, processes are needed that do not explicitly add value or define value, but which support the processes described above.

The Underlying Success Principles

Mostly nowadays organisation is nothing more than a resource pool still organised by functions and only supporting the processes by staffing projects. But what is needed is an organisational form with optimised for the software process and all other important processes within a company [4]. A virtual organisation is such a form. It must posess a few underlying principles that are necessary for its economic success. It must be [8]:

Process Oriented

Each process has a end-to-end responsibility to fulfil the contract to decrease the time-to-market and increase the performance all tasks for a particular project have to be organised as a process. The flow of information, work and products are inherently combined together.

Customer Oriented

Customer satisfaction is the primary goal for each process as well as for the virtual organisation as a whole. Therefore it is necessary that each software process could conceivably be a customer for other processes as well as a supplier for their own customers.

Transaction Oriented

Between software processes, there are order-delivery relationships with clearly defined interfaces for communication and data exchange. Through streamlining the amount of interfaces should be reduced since they are sources of information loss.

Object Oriented (principle of autonomy)

Each of these processes can be seen as an object that uses resources and fulfils tasks. Objects can be described via different performance and quality parameters. These parameters are combined with metrics that are used for the selection of the best available processes for a particular virtual organisation.

Continuous Improvement

In a virtual organisation, continuous improvement occurs at the network level because the network is constantly improving the co-ordination and communication between the individual companies (nodes). And at the individual node level, each company is constantly improving its core competencies.

Virtual

Companies have to go with their products and services to the geographical location of their customer, and they have to talk the same language as them. Distributed processes are necessary to fulfil such a requirement. A virtual organisation is the result of such an intention.

By combining the core competencies of many individual companies within the network, each virtual enterprise is more powerful and flexible than it individual parts.

Each company in a virtual organisation is chosen because of its process excellence. The result is a more powerful organisation since it is made up of the best available core competencies. By having all partners agree with and commit to defined schedules and costs before the start of the project, the risk is reduced to a minimum.

What is the Economic Impact

To stay competitive in today's global market, it is necessary to set up win-win based agreements in cost sharing projects where partners from different countries share the risk and the effort, and jointly exploit ideas, products, and services. Through effective and distributed collaborations, organisations can cut down their risk significantly (e.g. sharing the development cost with other partners), and can reach a much larger market.

This new approach of collaborative development leads to opportunities for creating financial leverage (by joint risk and effort funding) and an increased marketing leverage (by joint representation on the market, and larger distribution through a network of partnerships).

One of the principal benefits is that it provides strategies for improving service velocity - the rate at which software can be brought to market and/or customised. Through such an organisation, access to resources, know-how, and the markets of partners are available, in a way that costs and time can be saved in favour of a joint production. However, the major goal is, via an optimised information management, to increase flexibility, productivity and customer orientation [18]. Quality itself is not the primary goal of a virtual organisation, rather it is accepted as a necessary condition for maintaining competitiveness [23].

Collaboration through a virtual office also implies a need a need for a knowledge base that can be shared between organisations. The difference between information (as it is offered now by many Web servers) and knowledge is that knowledge is created from information by putting a structure onto the information so that it can be shared, multiplied, and understood across a team. Not only the information and its structure is relevant, but also meta-information like owner, creation time, or when the information was last retrieved and by whom. This information about information is also necessary for turning information into knowledge.

This may also have a strategic impact for the European Union in general. Under the 4th framework program the ESSI (European Systems and Software Initiative) initiative funded hundreds of PIEs (Process Improvement Experiments) at smaller and medium sized companies across Europe to improve their development capabilities and software processes.
The 5th framework program supports the virtual information society strategy so that technologies are sought that could connect those efficient companies into focused collaborations building the strengths together as if they are a big company.
In a Europe where most industry is small and medium sized, such a strategy could create competitive advantages against other countries and continents.

What are the Functions

Whereas the virtual organisation’s life span is limited by the software project’s life cycle, there is need for a permanent and preferable flat organisation that provides for the availability of required competence resources, optimised communication channels and definition of a meta process facilitating co-operation and interaction. Therefore, management activities encompassing all software processes are needed to configure the virtual organisation, to co-ordinate their co-operation, and to conform them to a common strategy.

Distributed collaboration requires effective co-ordination between the involved partners' work and quality control mechanisms. This can be addressed with by a virtual office on a network that includes project archives and document management, configuration management, guide-lines and computer support for project documentation, network and computer supported information flow, and appropriate security mechanisms assuring privacy of the materials exchanged and produced. In addition such a system needs a flexible access control and authentication system to manage the access to all information for every individual on the network.

However, virtual organisations are not limited to just such quality assurance functions, they can also offer a vast array of additional services like customer support, project management, component and other administrative functions, where quality assurance is a core component.

The majority of communication in a distributed collaboration uses asynchronous mechanisms (e.g. email, web-publishing and retrieval) but there is also need for synchronous communication like chat or telephone conversations. The information which is included in synchronous communication should also be archived by the information infrastructure. The integration of information- and telecommunication systems seems to be necessary.

Requirements for successfully applying the concept of a virtual organisation to the software production process are among other things an open and standardised information infrastructure, defined software processes, confidence in the performance of all partners involved, the participation of all partners in the decisions and the overall result, as well as a modular software architecture.

Individual companies can be geographically distributed and therefore use different languages, and different legal and social systems. The connecting information and communication technology is charged with selecting, measuring and controlling the processes in spite of these constraints. The underlying technologies have to meet the requirements for knowledge and information storage, for de-centralised information access and retrieval, as well as for the short-term merging of distributed knowledge [19].

Information Infrastructure

However, most of the requirements that virtual organisations demand (in terms of information infrastructure, especially technical openness, distributed storage of data, and security mechanisms) can be fulfilled with existing information and communication technologies [18]. Today, the most powerful and cost-effective infrastructure for enabling virtual organisations is the Internet [3]: with its failsafe network topology as the communication infrastructure and the different Internet services like email, WWW and ftp, as the information infrastructure.

Via virtual private networks virtual organisations can use the Internet as an Intranet. The information (e.g. project documentation, customer data) that is necessary for performing the business is presented by application servers to all the members of a virtual organisation. Such a Web-based application server is the Hyperwave Information Server - a broad and feature-rich application development platform that has been used to build web-based applications in areas that are important for virtual organisations: knowledge management, document management, Web-based training, project management, and many more. This flexibility and customisability is one of the major strengths of the Hyperwave Information Server.

To meet the requirements discussed above for a virtual organisation, flexible IT-systems are necessary. They have to be quickly adaptable (like the plug&play concept) to new processes and IT-system. As an example for such a system the Hyperwave Information Server supports a virtual organisation and its processes with a great deal of built-in functionality including the following:

A dynamic structure for the presentation of documents and links that allows customised views of information
A clear separation of information and its presentation
Documents are stored as objects containing information, metadata and functions
A built-in user and group based security mechanism (access control)
A scalable architecture for connecting many Hyperwave Information Server together into one server pool
A channel mechanism that supports passive information retrieval (notification)
Users can create new information through linking documents together
Collaborative authoring is supported with integrated document versioning, locking, and configuration management
Object-Oriented programming methods allow the development of new applications and/or the extension of existing functions

Combining the concept of an virtual organisation with such an infrastructure leads to following simplified scenario:

Each process (or company) runs its own information server for their special tasks. Process dependent applications control the input and output of data. All servers in an virtual organisation are combined into a server pool where each member can access information from the other. But some information should not be accessible to the whole virtual organisation. Therefore the access rights must be restricted to the information that is defined in the general agreements. These contracts also specify the overall workflow and the interfaces between processes. Information that should be exchanged is linked from the server that holds the information to the server where the information is needed without copying it. This reduces the amount of storage, increases the maintainability of information (only one source), and simplifies the access control. After a common project is finished, the servers are removed from the server pool and all links between servers are automatically removed or disabled.

The features and the scenario described above are only a small overview of what is possible when a Web-based server act as an information infrastructure for a virtual organisation. Better is a real world example of an actual implementation for a specific project. NQA is such an example. It supports the quality management in a virtual office in conjunction with ISO 9001.

NQA (Network based Quality Assurance)

Paradigms Underlying the NQA Concept

The NQA approach bases on three principles which have been discussed and published at previous ISCN conferences (http://www.iscn.ie/conferences) and about which there is a book being published by IEEE [13]: Better Software Practice for Business Benefit - Principles and Experience (ed. Richard Messnarz, ISCN).

The three principles

Role and information flow based team work process management
Development by configuration
Re-Use pool concept

are discussed below.

A major feature to make such a virtual approach applicable for different environments is that such a system must be kept completely configurable. The menu, the data, the functions, the document/information flows can be configured for different user scenarios and this high configurability is the major feature of an NQA virtual office. It is based on standard Internet languages and scripts and on the Hyperwave information server.

The Underlying Management Principle

A software process is not seen as just a sequence of tasks with a planned result [10], but it is the result of an integrated team work environment [14]. The organisation is broken down into work scenarios (management use cases, e.g. scenario for planning, scenario for design, scenario for marketing, etc.) and each scenario is designed with

Roles who have responsibilities
Work steps to which roles and resources are assigned
A network of work steps forming a work-flow
Results produced by roles performing a certain work step in the work flow

The new approach is to think role-centered, so that by staffing of roles work scenarios in an organisation are initiated.

The advantage of the new approach is

People know their responsibilities better and know their communication interfaces to other members in the team
New staff can easily be integrated (assign a role, learn the skills required to play the role, follow the communication flows in the team)
Information technologies like NQA (because the communication interfaces become visible) can be used to support the team communication, documentation, and configuration of results.

Benefits Measured

Experiments with this approach ave been carried out since 1993 at firms in Austria, Germany, Spain, and Ireland, and 7 other countries. Results are [13], [14]

A 50% reduction in effort in new staff integration
A 67% higher team motivation for using documentation efforts like ISO 9001 (share the work in a team in a defined way)
A 67% reduced maintenance and 50% higher productivity because a decomposed role based team with clear responsibilities allows good distribution of tasks (parallel and not sequential work) and avoids monolithic program architectures (all are responsible for the same software without clear distinction of interfaces and modules).

Management Steps

Define the roles
Identify communication flow between the roles
Formalise communication flows (only where necessary) and define results
(exchanged between roles)
The work-flow, after that, is just a waste product of the team-work model

Example from a planning scenario at Hyperwave

For each scenario there is an underlying role play clearly describing the roles played in a team, the responsibilities, and the communication flows. These communication flows result in a number of work instructions describing the roles' duties and the sequence of work steps to be performed. The same working instructions are then used, for instance, to show compliance with working instructions required by ISO 9001 [11], [12].

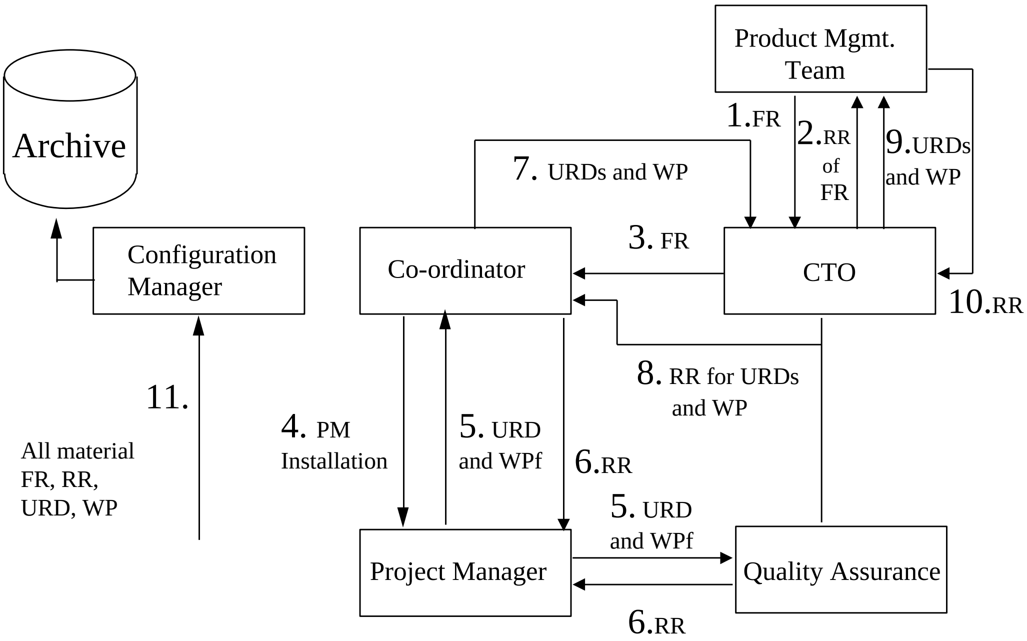


Figure 1: A Role Play for Feature Request Management and Planning

Work Instructions for the Feature Request and Planning Scenario at Hyperwave

The Product Management Team (PMT, customer) makes a Feature Request (FR). The Chief Technical Officer (CTO) receives it and archives it.
The CTO reviews the features together with the PMT resulting in Review Reports (RR) for the feature request and decisions about their implementation.
The refined feature request (for which an implementation was decided) is forwarded to the Co-ordinator (CRD).
The Co-ordinator assigns the feature request to a responsible project manager. For each release there are many such feature requests so that the previous steps are repeated many times.
The responsible Project Manager (PM) draws up a draft User Requirements Document (URD) and a URD specific Work Plan (WPprj), and forwards the draft for review to the Quality Assurance (QA) and the Co-ordinator (CRD).
The draft URD and WPprj are at the same time reviewed by the Quality Assurance (QA), and the Co-ordinator (CRD), resulting in Review Reports (RR).
The Co-ordinator approves the WPprj and combines them into an overall Work Plan (WP) for the organisation, and forwards all URDs and the overall WP for review to the CTO.
CTO approves the URDs and the WP.
PMT receives URDs and WP for final review.
PMT reviews and gives acceptance to the URDs and the overall WP.
Configuration Manager (CM) controls that all materials produced in the work flows have been properly archived. Special care is taken on the trace-ability between feature requests, requirements in the URD, and proposal/agreement issues.

Only after the establishment of such a role-based model the information flows become clear and a tool can start to support the team communication and quality control activities through a virtual office of distributed competence teams.

The Information Technology Principles Underlying an NQA Concept

Development by Configuration

This paradigm bases on the fact that functionality is to be separated from data, and that data can be assigned with functionality by the user through configuration. NQA concepts must developed according to this principle and allow each organisation to insert their own documentation or result templates, and the NQA system then automatically generates (with the creation of objects from the templates) the functionality to the created objects.
This way users can insert and maintain document or result templates and adapt the system to their own specific documentation requirements without any change or customization of code (just by configuration of data).

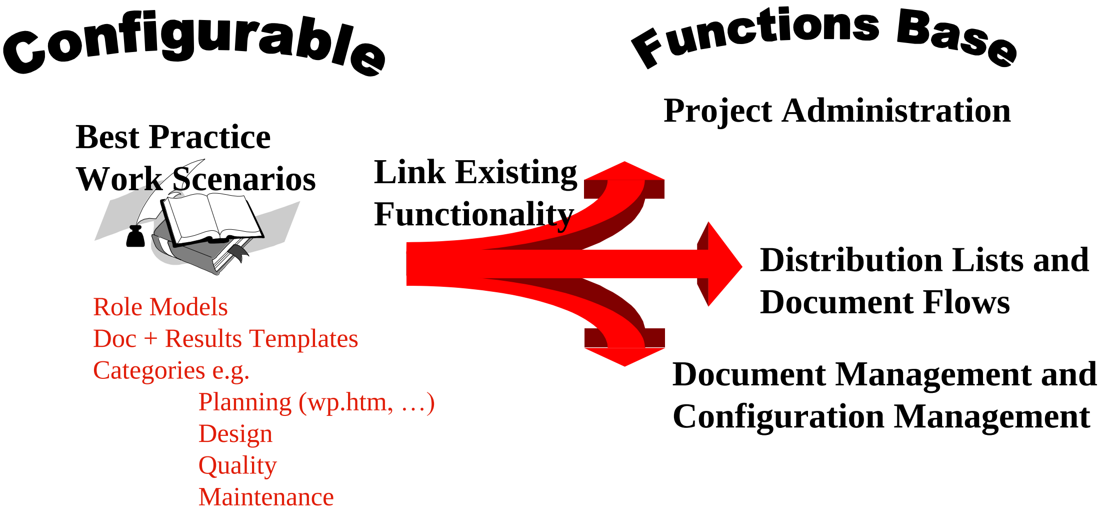
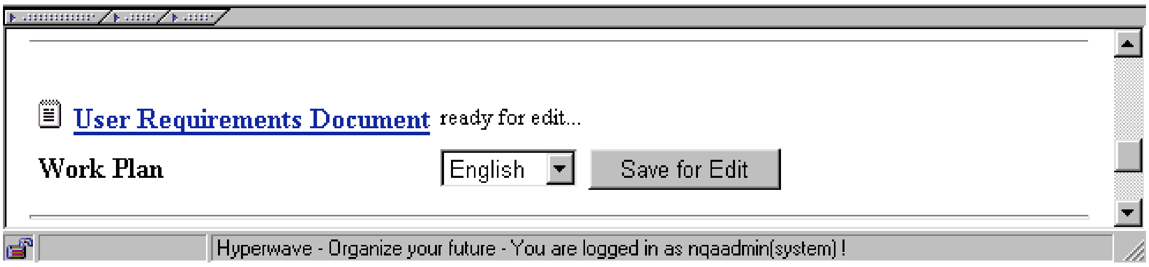


Figure 2: Data and Functional Configurability

Function Base Driven Configuration (Re-Use Pool Concept)

At the moment three basic elements can be configured to which the above functionality is generated.
Documents - The below picture shows the standard window for document creation, with SAVE the functionality is generated to the template taken from the pool and a first version is issued under configuration management)



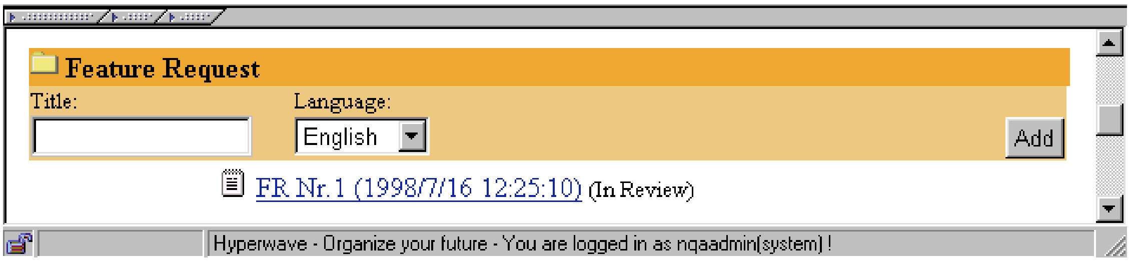
User Requirements Document ready for edit...

Work Plan — English — Save for Edit

Hyperwave - Organize your future - You are logged in as nqaadmin(system) !

Figure 3 : Document Object Creation Window

Reports - The below picture shows the standard window for report creation, after ADD a report is added to a list and the functionality is generated to the template taken from the pool and a first version is issued under configuration management.



Feature Request

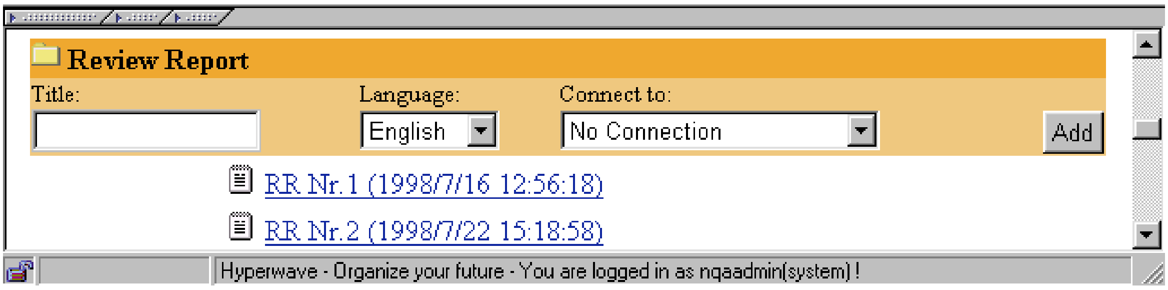
Title: — Language: English — Add

FR Nr.1 (1998/7/16 12:25:10) (In Review)

Hyperwave - Organize your future - You are logged in as nqaadmin(system) !

Figure 4 : Report Object Creation Window

Linked Reports - same as with reports, plus the report is automatically linked backward and forward to what has been selected in the right combo boxes.



Review Report

Title: — Language: English — Connect to: No Connection — Add

RR Nr.1 (1998/7/16 12:56:18)

RR Nr.2 (1998/7/22 15:18:58)

Hyperwave - Organize your future - You are logged in as nqaadmin(system) !

Figure 5 : Linked Report Object Creation Window

Depending on the user needs the three elements are configured. E.g. Linking Feature Requests (FR) with user Requirements Documents (URD), so that an URD is automatically created by the links to accepted FRs (example from a customer wish from Daimler Benz).

Further basic elements might be considered and inserted into the NQA configuration pool in later releases.

How an NQA Virtual Office Works

A required functionality of an NQA system comprises the automatic assignment of the following functionality to created objects -

Document Management

Creation of documents from a template pool (configurable by customer). Automatic administration within a project structure under a certain documentation category (e.g. planning document). Electronic submission to a distribution list (workflow). Version management and change control (see configuration management). Automatic forward linking to reports (e.g. a number of Review Reports linked forward and back to the document version, see link management). Download, edit, and publication facilities. Computer supported test status.

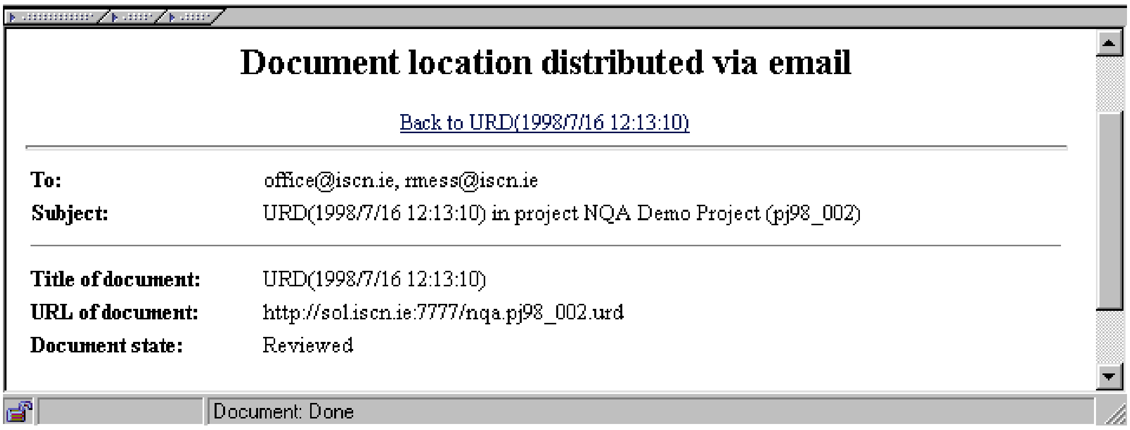
Report Management

Creation of reports from a template pool (configurable by customer). Automatic administration within a project structure under a certain documentation category (e.g. quality control reports). Electronic submission to a distribution list (workflow). Version management and change control (see configuration management). Automatic forward and backward linking between documents and reports, or reports and reports (e.g. linking test protocols with problem reports, and problem reports with modification reports). On-line edit of forms at server side, and on-line submission (no download necessary for edit).

Workflow Management

Electronic submission of reports ad results to team members. Encryption module can be used. Administration of distribution lists (for automatic forward). A communication log per project archiving all communication flows between team members (roles).

People are assigned to project groups, and distribution lists are automatically generated and proposed for the submit of documents and reports.



Document location distributed via email

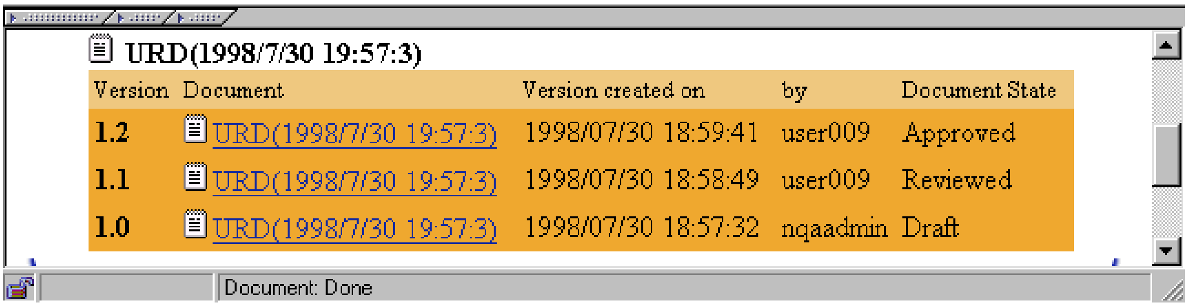
Back to URD(1998/7/16 12:13:10)

To:	office@iscn.ie, rmess@iscn.ie
Subject:	URD(1998/7/16 12:13:10) in project NQA Demo Project (pj98_002)
Title of document:	URD(1998/7/16 12:13:10)
URL of document:	http://sol.iscn.ie:7777/nqa.pj98_002.urd
Document state:	Reviewed

Figure 6: A Standard Notification Message for Submissions

Configuration Management

Version management. Registration of versions in a document (result) history. Check-in and Check-out functions. Revert to previously archived versions. Test status information in document history.



URD(1998/7/30 19:57:3)

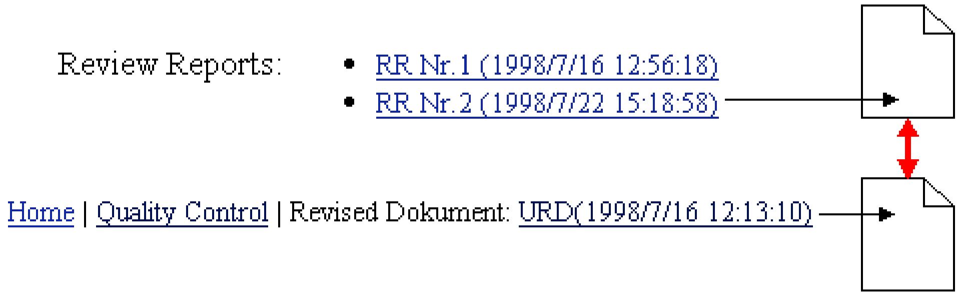
Version	Document	Version created on	by	Document State
1.2	URD(1998/7/30 19:57:3)	1998/07/30 18:59:41	user009	Approved
1.1	URD(1998/7/30 19:57:3)	1998/07/30 18:58:49	user009	Reviewed
1.0	URD(1998/7/30 19:57:3)	1998/07/30 18:57:32	nqaadmin	Draft

Document: Done

Figure 7: Version control with Object History Including Test Status

Link Management (Forward and Backward Tracing)

Definition (see functional configurability) of links between report and document types. Automatic assignment of linking properties to created objects. Automatic forward and backward linking according to the defined functional configuration (configurable by system administrator). E.g. linking review Reports with documents, so that by a click you switch between the document and the related reports.



Review Reports:

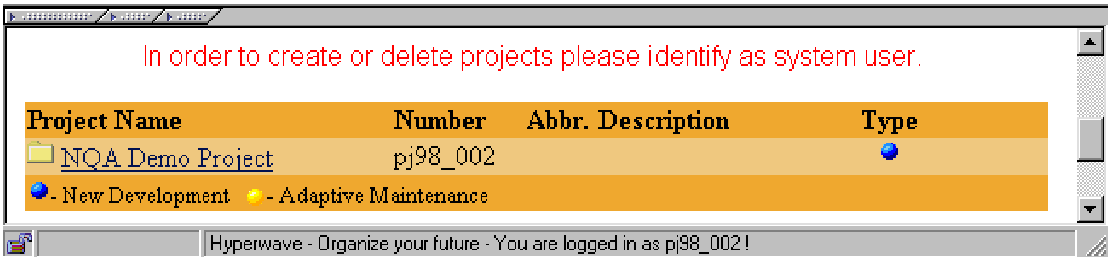
- RR Nr. 1 (1998/7/16 12:56:18)
- RR Nr. 2 (1998/7/22 15:18:58)

Home | Quality Control | Revised Dokument: URD(1998/7/16 12:13:10)

Figure 8: Forward and Backward Linking (e.g. Review Reports Linked to a User Requirements Document)

User Administration

Administration of a team per project (see only their project). An NQA system administrator nqaadmin (sees all). Administration of a distribution list (for electronic submission) per team (and in future role based per document).



In order to create or delete projects please identify as system user.

Project Name	Number	Abbr. Description	Type
NQA Demo Project	pj98_002		●

● - New Development ● - Adaptive Maintenance

Hyperwave - Organize your future - You are logged in as pj98_002 !

Figure 9: Identification and User Control

Security Management

No access without identification possible. The information in the electronic submissions contains only links to info at the server which requires identification. If even these links should be protected an additional encryption module from Hyperwave can bee installed.
By just using Netscape the team members (from home, from any work place, etc.) can access the NQA virtual server and work on-line through a joint interface.

The ISO 9001 Experience Pool

NQA is delivered together with a complete electronic ISO 9001 manual, with examples and role plays and with templates in German and English for all required ISO 9001 documentation.
It is also delivered with implemented procedures and scenarios to run ISO 9001 compliant planning, design, delivery, and maintenance.

This comprehensive set of information allows organisations to achieve an ISO 9001 certificate much easier.

It also gives (through its configurability of a template pool) organisations who have already an ISO 9001 certificate a huge support in shifting from a paper based ISO 9001 environments to a fully electronic based ISO 9001 computer supported system,.

NQA systems at organisations in Austria, Germany, and Hungary have already been certified by TÜV, Norske Veritas, and ÖQS.

NQA's Future Plans

NQA is an application developed by ISCN on top of the Hyperwave information server. It is planned that NQA in future is encapsualted as a virtual quality assurance solution for ISO 9001 under Hyperwave, and sold on a CD to all Hyperwave users.

Future Outlook into Virtual Organisations

As an organisation form the co-operation network virtual organisation is very suitable for the development of software [3]: not only to decrease costs, to be able to react to rapidly changing situations in a flexible way and to distribute risks, but moreover to increase customer benefits by ensuring performance and quality. To achieve this, it is necessary to understand the underlying methods and processes, especially because of the already existing quality problems in software development [4][9].

The quality of the whole company is a condition for the quality of the products they produce - the quality of the processes and the quality orientation of the company culture, as well as the quality of the employees and the management [16]. Because of the temporary nature of the co-operation no identification with the virtual organisation will evolve, and therefore no company culture as well. The loyalty to the company will be replaced by the loyalty to the product.

It is also a disadvantage that the performance of the Internet is not as good as it should be, because of insecure parts of networks and slow transmission rates. A final area of problems is represented by the existing legal vacuum, especially in terms of the validity of legal documents in electronic form, the acceptance of electronic signatures, patent rights and international product liability [18].

References

[1] Arnold, O., Faisst, W., Härtling, M., Sieber, P., Virtuelle Unternehmen als Unternehmenstyp der Zukunft?. HMD – Handbuch der Datenverarbeitung, 32 185/1995, pp.8-25.

[2] Cusumano, M., Japan's Software Factories: A challenge to U.S. management. Oxford University Press, New York, Oxford, 1991.

[3] Gao, J.Z., Chen, C., Toyoshima, Y., Leung, D.K., Engineering on the Internet for Global Software Production, IEEE Computer, May 1999, p.38-47.

[4] Gillies, A.C., Software Quality: Theory and management. Int. Thomson Computer Press, London, 1992.

[5] Hammer, M., Champy, J., Reengineering the Corporation. Harper Collins Publishers, New York, 1993.

[6] Handy, Ch., Trust and the Virtual Organization. Harvard Business Review, May-June 1995, pp.40-50.

[7] Herzwurm, G., Mellis, W., Schmolling, K., Software Factory - Ein Statusbericht. HMD – Handbuch der Datenverarbeitung, 180/1995 p.8ff.

[8] HPO, High Performance Organizations Handbook. Technical University Graz, 1996.

[9] Humphrey, W.S., Managing the Software Process. Addison-Wesley, Reading, 1989.

[10] IEEE *Software Engineering Standards Collection*, IEEE Standards for Software Quality Assurance Plans (IEEE 730-1989), Quality Assurance Planning (IEEE 983-1986), Project Management Plans (IEEE 1058.1-1987), Configuration

management Plans (IEEE 828-1990), Software Verification and Validation Plans (IEEE 1012-1986), IEEE Computer Society Press, 1991

[11] ISO 9000-3. Quality management and quality assurance standards. International Standard. Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software. ISO (1990).

[12] ISO 9001. Quality Systems. Model for Quality Assurance in Design/ Development, Production, Installation and Servicing. International Organisation for Standardisation, Geneva (1987)

[13] Messnarz R., Tully C. (eds.), The PICO - Book*: Better Software Practice for Business Benefit - Principles and Experience*, IEEE Computer Society Press, in publication

[14] Messnarz R., Stubenrauch R., Melcher M., Bernhard R., Network Based Quality Assurance, in: Proceedings of the 6th European Conference on Quality Assurance, 10-12 April 1999, Vienna , Austria

[15] Marciniak, J.J., Reifer, D.J., Software Acquisition Management. John Wiley & Sons, New York, 1990.

[16] Mellis, W. Herzwurm G., Qualitätsmanagement. Business Computing, 8/1994, p.36ff.

[17] Mellis, W. Herzwurm G., Stelzer, D., TQM der Softwareentwicklung. Vieweg, Wiesbaden, 1996.

[18] Merkle, M., Virtuelle Organisationen – Ihr Erfolgspotential: eine integrative Informationsinfrastruktur. Institutsbericht des IFI der Universität Zürich, 1996.

[19] Mertens, P., Faisst, W., Virtuelle Unternehmen, eine Organisationsstruktur für die Zukunft?“. Technologie & Management, 44 2/1995, p.61ff.

[20] Mertens, P., Faisst, W., Virtuelle Unternehmen nutzen weltweite Netze, 1996.

[21] Picot, A., Reichwald, R., Wigand, R.T., Die grenzenlose Unternehmung – Information, Organisation und Management, 2. Auflage. Gabler, Wiesbaden, 1996.

[22] Scholz, Ch., Die virtuelle Organisation als Strukturkonzept der Zukunft? Arbeitsbericht Nr.30, Lehrstuhl für Betriebswirtschaftslehre, Saarbrücken, Universität des Saarlandes, 1994.

[23] Tapscott, D., The Digital Economy. McGraw-Hill, New York, 1996.

[24] Zesar, K.D., Zechner, M., Salhofer, P., Schuster, G., Muelleitner, G., Performance and Quality Aspects of Virtual Software Enterprises. In: Proc. of the 24th EUROMICRO Conference, Västeras, Sweden, August 1998.

[25] Zesar K.D., Mesaric G., A Process-Oriented Quality Management Model for Software Developing Cooperation Networks. Proc. of the 6th European Conference on Software Quality, Vienna, April 1999.

Results from the ESSI process improvement experiment -Virtual Team

Author : Jochen Lüling

In the past teleworking was used only in big companies. This was a result of the high costs for the technical infrastructure. Due to the new technologies like the internet, teleworking is becoming now efficient also in small companies or small teams. The IVM AUTOMOTIVE Stuttgart GmbH has executed a process improvement experiment called "Virtual Team". In this experiment StarTeam (a software configuration management tool), remote access and videoconferencing was tested to establish a new software development process.
The chosen small budget solution was tested in two baseline projects. The results of these projects show, that it is possible to make effective and economic teleworking even in small teams.

1. Summary

1.1 VITE (virtual team)

IVM AUTOMOTIVE
Stuttgart GmbH
Blumenstraße 29
70736 Fellbach
Telefon
0711 / 9514-0
Telefax
0711 / 514026
Internet
http://www.ivm.com
eMail
stuttgart@ivm.com

IVM provides specialist solutions, mainly to the automotive industry. The total IVM group has more than 2000 employees. The subsidiary at Stuttgart employs a staff of 50 in its software development team. Due to the fact that most of the software produced by IVM is custom software built in accordance with requirements specified by the customer, the developer are sometimes required to work on customer sites, wherever these may be. Other development resources are scattered through the company. In addition there is an increasing part of external consultancy assistance.

The purpose of IVM's PIE, VITE (Virtual Team) is to put the technical infrastructure and processes in place to support geographically distributed working. This will have three main benefits :·

It will allow employees to work from home or from customer sites, while still remaining part of a development team. This will cut down time wasted in travel and improve their quality of life.
It will allow IVM to make better use of the specialist resources scattered across the company. Before the PIE, there was little knowledge transfer between software engineers in different parts of the company.
It will enable IVM to bring in contract workers/partners on specific projects where there expertise is needed, for example, it envisages closer ties in the future with partner organisations in Germany or India.

The Internet-based technical infrastructure was deliberately chosen to ensure that it could be used by small project teams on low budgets. On the process side, VITE has had to carry out a number of adaptations to its ISO 9001 procedures in order to support distributed working. It identified process requirements by interviewing project leaders and developers using a 'Bootstrap-like' questionnaire with 60 questions on topics ranging from quality management and metrics to organisational capabilities and employees' knowledge. The PIE team then set about putting in place tools and structure to support the extended processes, for example,

New StarTeam configuration management tool in place of SourceSafe, because StarTeam has a Internet browser interface.
New firewall concept

The new processes and infrastructure are being tested on two baseline projects developing different systems in different languages. The aim of this exercise is to come up with a generic set of rules and guidelines to be used in distributed development. Once IVM has established common procedures, processes and directory across the development organisation, it will be easier to incorporate external consultants into teams, and to use resources flexibly between teams. The PIE has come up with a useful checklist of organisational issues that teleworkers need to consider before they can start in this role, for example: have they got appropriate home insurance, the right technical infrastructure, call forwarding to their home etc. . The chosen solution is for a development team of about 50 employees. Therefor it can be easy adopted to other companies where the number of involved staff is ca. 40-80 employees.

1.2 Content of presentation

the technical part of the VITE experiment e.g.
The tools evaluated, StarTeam, NT RAS Server, PictureTel LiveLan
The technical solution and the firewall concept
The new process enhancements for the distributed development e.g.
New development process due to the versioning and change requests via the internet
New checklists / flowcharts for creating a teleworkers place
New contracts with the employees
The results of the PIE e.g.
An increase of 20 % of the distributed development
the lessons learned during the experiment

1.3 Information about the speaker Jochen Lüling

Born 1965, Stuttgart Germany
Academically qualified mechanical engineer 1992
Working at TechnoData 1992-1997
Working at IVM since 1997
Microsoft Certified Solution developer 1997
IVM project leader 1998
IVM team leader 1999

At the moment he is responsible for several projects developed in Visual C++ and Visual Basic for Oracle or SQL-Server databases.

Starting Scenario

Experiment context

Based on experiences we became convinced, that there must be a better more flexible way to use our specialists located in different teams at different subsidiaries.
Together with the requirement of the former very rarely used possibility to work together with external employees we created the project virtual team (VITE).

Two goals should be achieved with the new possibilities of the virtual team:
Optimising the creation of proposals with assistance of external specialists
Greater flexibility at the implementation phase e.g. with more possibilities of choice of resource

The experiment context is divided into two parts :

Technical context:

A new access from outside IVM using new technical features like videoconferencing or internet connectivity. Through remote access all employees should have the total functionality of the local network like the internet services SMTP, HTTP, FTP, NNTP and DNS. In addition customers will have the possibility to work with the HTTP based software configuration management tool over the internet.
It is important to take into consideration the fact that IVM is a company with 2000 employees world-wide but the software development group of the subsidiary of Stuttgart has only about 50 members. Therefore the solution is interesting for a small company or development team.
It is a low budget solution: This means the videoconference system is a desktop system, the firewall is not build with expensive special router hardware but with a cheaper software solution and good but nevertheless “standard “ ISDN cards.

Adoption of the software development process:

The origin of the experiment resulting in the purpose of the virtual team is not originated in a deep diagnosis of the organisation against models like bootstrap. It is more the result of some bad experiences we made. Therefore at the beginning of the experiment a deep analysis of the starting scenario is important to focus the purpose and to have later a metric for the success of the project.
An analyse and evaluation of
the discussions and interviews with the project leaders and team leaders of the baseline project
A questionnaire (sort of “small bootstrap”) fill in through the hole development staff
will bring us the results.

With enhancements and optimising of the existing process definitions for software development, the new distributes work will be well co-ordinated and documented.
An example of a change is the switch in the software configuration management utility (SCM). We used former Microsoft visual source safe. The new tool we use is called StarTeam from Starbase. This offers a lot of enhancements e.g. the possibility to check in and out files over the internet. This is an important criteria for our new environment.
The new process requires the adoption of our existing ISO process definitions. The steps in forms like flow charts to build up a new teleworking place (e.g. at the developers home) must be created.

The size of the development team of the 2 baseline projects is 8 employees.
The baseline projects will be used to test the new communications and data exchanging methods with the customer. The team members are high qualified software engineers e.g. we have Microsoft certified solution developers. We have no stand-alone quality team inside of the development group. At the moment the quality actions are performed by the team itself, which is not easy if involved in a

productive process. A purpose of the experiment is to bring together the rules and guidelines of the teams resulting in a company guideline.

Company Context

IVM was founded 1968 and the company grew from a original small local company, called Engineering office for Process technique and Engine design, into an international corporation. That means one has to reach the quality and productivity required on the international market. That goal was reached in small steps, for example the German company group is ISO 9001 certified. With the next step we will try to put resources, that are located in different offices, to better use. That target should be reached by establishing requirements for distributed development.

Right now a small team at any IVM office is always responsible for his software development. Communication between offices takes place only twice a year at internal workshops. It is currently impossible to make use of available resources easily, when a difficult situation arises within a project. This is unfortunate because the various IVM offices posses a couple of software specialities. Even if a project could use the assistance of a specialist, distance and limited telephone communication makes this difficult.

Teleworking at home was not possible until now. The important integration in the normal working process lacks.

External consultants are used rarely because integration into the development environment is difficult and inefficient. An easier communication with consultants, defined access to the sources, and well-defined programming guidelines should bring improvements.

As mentioned above, most IVM software is customized software developed in response to specific customer requirements. These projects involve the use of many different tools. At the moment we have projects being developed with Visual Basic, Access, Visual C++, C, LabWindows and Java. The long training period for new projects is not just a result of the specific project knowledge demand and the different tools, but through the different quality assurance rules between the projects or tools. Quality control depends on the project requirements and is performed individually. Some programming rules exist but they are language dependent. Developers tend to specialise in a language and this reduces flexibility with respect to taking action on behalf of different projects.

Experience has shown that our use of the special skills available throughout our organisation is poor at best. More effective use of all resources has been identified as an important corrective action by our management. This means that we must create the technical and organisational skills to be able to get easy access to distributed resources.

3. Experiment description

Building up on the analysis of the existing quality assurance rules, style guides and programme rules have been improved with the aim to be project and language-independent. Distributed development requires more enhancements of the normal ISO processes. We will strive to solve problems with co-ordination, information or team-development in a suitable process.
As a baseline, two projects for different customers were chosen, developed in different languages at various locations.
The Internet will be used for access to sources for the project. Therefore it is possible to do version control, defect tracking, threaded management, audit logs, and similar work without regard to the location of the developer.
The new possibilities must conform with a high security requirement. Therefor a detailed tool and system evaluation took place.
The key for a effective distributed development is the way you can communicate with your partners. To achieve a high level, IVM will use a videoconference system, which offers not only the possibility to see and speak with each other, but also to work on different screens with one application.

Evaluated tools

Video Conference :

The tool evaluation has shown, that in the segment of videoconference systems a great bandwidth of quality exists. This is true especially for the desktop segment than for the room systems. An important criterion was the feature of application sharing. (a detailed description of the tool evaluation is shown in the annex)

On the basis of the evaluation the choice was LiveLan from PictureTel.

One videoconference system costs about 1000 ECU. If working with a server software and a gateway between then LAN and ISDN additional costs of about 2800 ECU arise .

The using of the videoconference system was very easy and comfortable:
The installation and using of the software was better than expected. Especially the application sharing possibility has been proven very well.

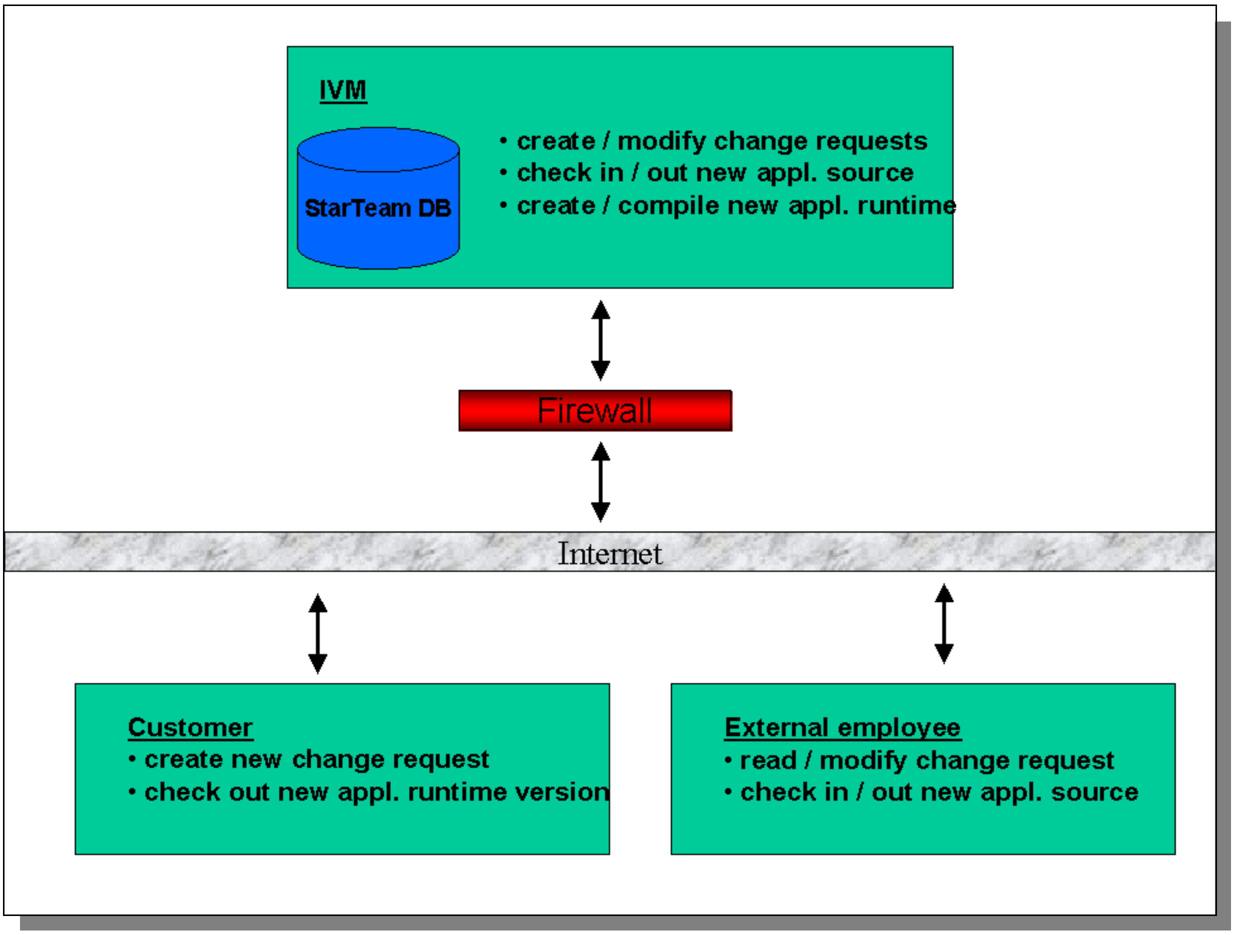
The following example is a typical scenario:
The developer at home has a software problem. He calls the specialist at IVM with the videoconferencing system. Because of the visual connection the team feeling is OK. The specialist at IVM is now able to open and debug the programme at the developer PC at home via remote control. For example look in the source code, try to reproduce errors etc. .

An important criteria is the ISDN connection. In the beginning of the baseline projects, it has been shown that there are at the customer site often no full ISDN port with an S0 bus, because normally the telephone system does not path the S0 bus. At this side additional costs for special cards in the telephone system would arise. This is in a bigger company in praxis very difficult to realise.

Because of this reason the internet connection is often the better connectivity possibility. This is a standard which is normally found at an engineer workplace.

Software Configuration Management tool : StarTeam

The chosen SCM tool offers new features :
- access over the internet
- change request management
- Mail integration in project management


IVM
StarTeam DB
• create / modify change requests
• check in / out new appl. source
• create / compile new appl. runtime
Firewall
Internet
Customer
• create new change request
• check out new appl. runtime version
External employee
• read / modify change request
• check in / out new appl. source

RAS Connection :

The new installed RAS Server is a NT-Server with 3 ISDN cards. They are connected with an 2x3 port ISDN connection which are working with one number. It is important that they work with analogue, ISDN and GSM calls. Through this RAS Server the developer at home reaches the hole LAN functionality.
Step 1 was realised with call back procedure. Therefor only special sites with explicit telephone numbers can use the RAS system. To enable more flexibility for example for the sales department, we are testing at the moment security token cards, to generate a dynamic password login.

The firewall concept for the e-mail, and internet connection has three stages :
- Screening Router outside
- Bastian host on basis Microsoft Proxy
- Screening Router inside

This concept brings time if the worst case happen and one stage is hacked. The screening router outside does packet filtering, the bastian host does IP masquerading and IP-shadowing .

4. Resulting scenario

4.1 Technical impact

The intensive work on the firewall theme has brought us a lot of know how. As an important factor we found a couple of security holes. Our old internet access was with no firewall protected (everybody thought that the internet provider has a firewall!) some unsecured modems were found etc. Even if we learned a lot with security, TCP/IP ... , we ordered additional training in this sector.
Some unsecured connections to our customers are redefined.

The new version control procedures are developed and implemented in the new framework of our production process. This was very important because less than 30% of a developer's time is spent in programming **new** software. Most of the time is occupied with existing code. Certainly in team development, time and money could easily be wasted on redundant efforts, accidentally overwritten code, etc.
Introducing StarTeam and the procedures takes on the time consuming and non-creative tasks of project management and version tracking, leaving developers free to concentrate on code. Ultimately, it also leads to better quality software, so developers spend fewer resources on code maintenance duties.

4.2 Business impact

An increase in productivity was achieved because we were able to use resources and employees effectively from geographically different departments throughout IVM projects.

For the identification of the financial impact of the virtual team for IVM we decided to use a metric system from Markus Forschner.
In the deliverable 2 we introduced this system in detail. In short terms the basic ideas are declared below:

The names of the 4 important process parameters of the top level are used like the suggested terms from “Forschner” (the perspectives of the balanced scoregards)

PP1 (1): *Financial strength*

(Kaplan/Norton: „The financial strength“)

PP2 (1): *Internal Effectiveness*

(Kaplan/Norton: „The internal process perspective“)

PP3 (1): *Customer benefit*

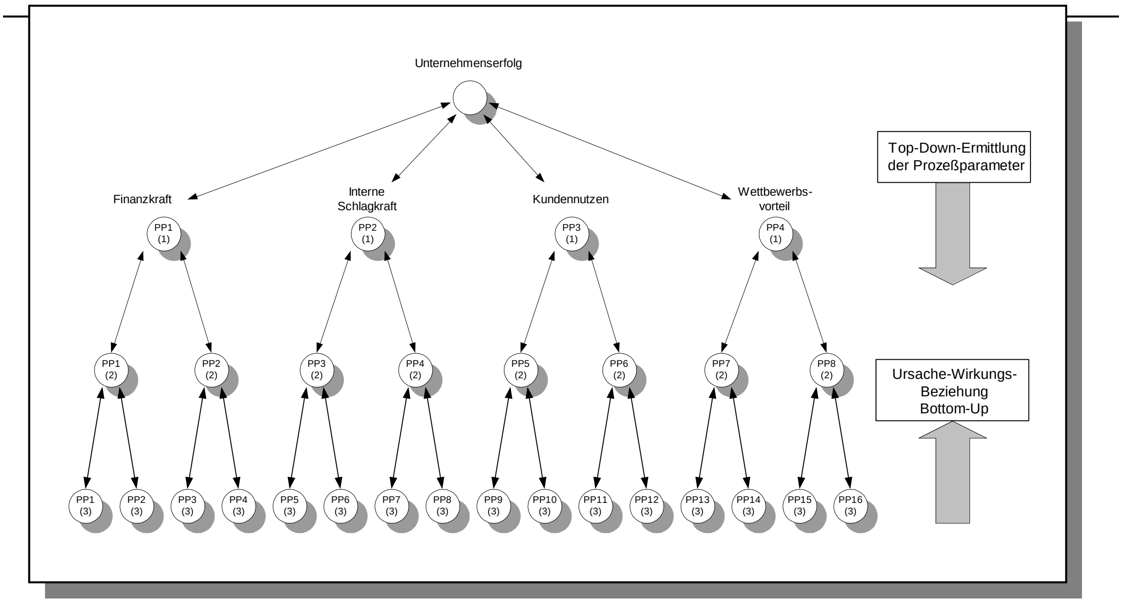
(Kaplan/Norton: „The customer perspective“)

PP4 (1): *competitive advantage*

(Kaplan/Norton: „The learn and development perspective“)

The impact of a process parameter of a process level is investigated with the process of hierarchy Fuzzy-Interferenz.

The process parameters of the level n depend on the process parameters of the level n –1 .



Based on this system we created for the four important process parameters a hierarchy chart. One example is shown in the annex.

The possible grading categories for each process are

-2 = much less
–1 = less
0 = equal
+1 = more
+2 = much more

The result of this metric for the Virtual Team is

PP1 (1): *Financial strength*	=	+ 0,7
PP2 (1): Internal Effectiveness	=	+1,3
PP3 (1): Customer benefit	=	+1,3
PP4 (1): competitive advantage	=	+1,3

The number of external subcontractors and external employees is rising. The good economical situation requires this flexible use of those resources. At the moment we have two new contracts with subcontractors at Baden Württemberg. This external assistance will help us to lower the internal pressure because of the many projects we have to handle at the moment.
Because of the new contract with one subcontractor we were able to get one order from our customer, which we had normally refused.

We are able to produce the product faster than before the PIE. This “time-to-market” aspect is an important benefit of the PIE.

Organisation impact

New definitions for installing a teleworking place have been created. An example is the new directory structure which must be used from all projects within the IVM subsidiary Stuttgart. The rising team work with other subsidiaries leads to the wish that they have to use this as well. This is discussed at the moment of writing this report.

Resulting from an analyse of the threat, a new security strategy was created.

The new security guidelines are sometimes not comfortable. For example we physically cut the connection to one customer. To work with this customer we have one PC standing stand alone. Even if not comfortable those actions are accepted.

It is important that the new tools and methods do not replace good know how for
– Operating systems
– network protocols
– risks'
– used applications

Security is not a unique action, but a continuous process.

4.4 Culture impact

The starting scenario for installing a firewall was not good. The developer had all possibilities which means all protocols for the internet : SMTP, HTTP, FTP, NNTP ... . In the first step after the installation of the firewall nothing was allowed and therefor the protest was great. Step by step were important protocols allowed, and in some longer discussions some other not. Every protocol which passes the firewall can be a risk. E.g. FTP was not allowed. The discussions lead to an expansion of consciousness concerning the security problems inside the team. In addition some hacking examples (e.g. sending an e-mail with an alien sender's address) show the risks.

VITE has brought together people of different nature: managers, senior engineers, developers. The combination of the business view, the solution oriented view and the theoretical view initiated fruitful discussions that resulted to a common understanding of the project's goals. Our effort to enhance our software development process with "tele-development-possibilities" caused a complete new cultural and organisational approach.

4.5 Skills impact

Through the intensive working on the security theme and the connectivity problems, the network know how was rising. The security awareness of the hole staff is now higher than at the beginning of the PIE.
A second important improvement is the working together of different project members. An example are the new C++ styleguides which are created from different teams.
A lot of discussions within the project improved the social capability of the team-members .

5. Key Lessons learned

5.1 Technological point of view

From a technological point of view, we learnt the following lessons:

In the project VITE , the firewall is an important part of the new distributed development process. Only if the firewall is secure against all attacks from outside, we can allow the developers to work in virtual team. The technique to configure , install and test the different firewall components is more complex than estimated in the Project proposal. As an example, all computers, our Mail Server, database server etc. had to be changed to a new TCP / IP address. Therefor a lot of problems arose which took more time than estimated. The security concept which divides the IVM network in different IP subnets is difficult to handle e.g. a lot of routing table entries had to be done. For this work good network and TCP / IP know how is essential and were sometimes missed. The planned effort of an external consultant was underestimated. Additional training for our network administrators is planned. Nevertheless the solutions show that a low budget solution is realisable and working fine.

StarTeam is accepted very well from the staff. It has an easy to learn graphical interface. It offers more possibilities than we are using.

The big problem we have in the baseline projects are
- the missing ISDN telephone ports (S0) at the customer site
- the high telephone costs from India

If we have no ISDN connection an important part : the videoconference system and the RAS connection is missing.

5.2 Business point of view

Right now we have a lot of work, new projects with short delivery dates. For those project we use with growing amount external employees. For the integration of those employees we use the new technology and process definitions.

The ESSI project is a very good marketing argument. At various seller’s meetings this process improvement experiment was introduced. The security actions (sometimes with a decline of comfort for the customer) were also positive accepted. The internet based process of change requests is very well accepted from the customer and is one of the most important impact of this PIE.

Training people in the new procedures has been an important part of this PIE. Training will be an ongoing task in IVM as procedures evolve.

References

Teleworking

http://www.telearbeit.de

http://www.telework.at/

LanLine Number 10/99 ,' Telearbeit und Remote Access' , AWI Lanline Verlagsgesellschaft

LanLine Number 11/98 ,' Telearbeit rechnet sich' , AWI Lanline Verlagsgesellschaft

Technologie&Management 4/97 ,'Anlagenbauer setzen auf Tele-Engineering'

SCM

http://www.teamproductivity.com/

http://www.starbase.com

Video

http://www.colibri.de/

http://www.man.ac.uk/MVC//SIMA/video1/toc.html

http://www.intel.com/proshare/conferencing/products/21data.htm

http://picturephone.com/

http://www.vcon.com/

http://www.videoconference.com/glossary.htm

http://www.vtel.com/

http://www.picturetel.com/

LanLine Number 12/98 ,' Videokonferenzen im LAN und WAN' , AWI Lanline Verlagsgesellschaft

EU :

http://www.cordis.lu/esprit/home.html

http://www.cordis.lu/esprit/src/essi.htm

http://www.cordis.lu/esprit/src/stessi.htm

http://www.esi.es/VASIE/Misc/ESSIgram/Download.html

Metrics :

http://irb.cs.uni-magdeburg.de/sw-eng/us/

http://irb.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/

http://www.mccabe.com

http://www.plumhall.com

http://www.iplbath.com

Ebenda S 49 ff., Traeger, Dirk: Einführung in die Fuzzy-Logik, 2. Aufl., Stuttgart 1994

Forschner, Markus:
Prozeßorientiertes Investitionscontrolling: Bewertung von Informationssystemen mit Hilfe der Fuzzy Logic, Schriftenreihe: Forschungs-/Entwicklungs-/Innovations-Management, Diss. Stuttgart 1996, Wiesbaden 1998

Kaplan, Robert S. / Norton, David P. (Hrsg.):
Balanced scorecard: Strategien erfolgreich umsetzen, Stuttgart 1997

Juran, Joseph M.:
Der neue Juran: Qualität von Anfang an, Landsberg/Lech 1993, S. 36 f.

Automating the Use of a Quality System

Oscar Tejedor-Zorita
GMV, S.A.
c/Isaac Newton, 11. P.T.M.
Tres Cantos 28760 Madrid
Spain
Tlf.: +34 91 807 21 00
Fax: +34 91 807 21 99
Email: otejedor@gmv.es

Introduction

GMV maintains since its origins a complete compromise with the quality of its services and products through the application of an ISO-9001 compliant Quality System (QS).

The QS was initially paper oriented and shows because of that a number of known weakness: Time consuming activity (both from staff and the quality manager), low flexibility (both in application of the procedures and updating the procedures themselves), etc.

To address this topic, some corrective actions were evaluated by GMV, and a Process Improvement Experiment (PIE) called ACQUASY (Automated Corporate Quality System) was defined to improve the efficiency and effectiveness of the GMV QS. The objective of the experiment is a system able to automate some of the tasks of the QS such as document search and retrieval, quality procedures application and measures collection and processing.

Motivations and Objectives

GMV S.A is a Spanish company which supplies engineering, consultancy and software services to a variety of international customers mainly in the aerospace market. The company was created in 1984 with the aim of providing engineering and consultancy studies for the European Space Agency (ESA). In 1988 the company was recognised as a Centre of Excellence in Orbital Mechanics because of the quality of its services. Since then, and under a continuous evolution, GMV has become more and more a software intensive company. This trend represents today that about 110 employees of the company out of 140 are mainly involved in software production.

The SW applications for the aerospace market must satisfy strict quality constraints (reliability, safety, etc..) and be under a permanent and strict control. Therefore GMV's offer must be characterised by the application of common advanced technologies and methodologies to satisfy the most demanding customer requirements. In other words, GMV's philosophy must be based upon the *mastering of technologies and methods*, which are afterwards applied in *this market where they can bring competitive advantages to the client.*

With customers demanding greater reliability and cost effectiveness, the company has continued improving its productivity and quality in its coped niche of activity (mainly ground and on-board space software). As an aerospace company, GMV started applying ESA PSS-05 standards for its developments. In 1985, these standards become the baseline reference for the company, and up to date they continue being used. This has claimed for a further effort aimed at gaining a tighter control of the projects being developed by the company. With the target in mind of maintaining its compromise with the quality of the services, GMV decided in 1995 to set up an ISO-9001 compliant Quality System (QS). After 18 months of quality procedure analysis and formalisation the GMV Quality System was finally put under application. This QS passed through several internal and external audits which put in evidence its drawbacks and which have been corrected during the application of the QS. Finally it will be audited for certification in the last quarter of 1997. However it is recognised by the company that this QS is, up to date, paper oriented and shows a number of known weaknesses. The risk is consequently that, on the long run, the QS may have some non-positive effects on the company productivity. Management of the identified risk was the main motivation for defining, developing and using the ACQUASY System.

To address this problem the following technical objectives were identified:

Facilitate the access to the QS documentation. Provide on-line access to the overall Quality Manual. This implies not just to support it by electronic means but to organise the documentation in such a way that can be indexed by different keys.

Facilitate the continuous adaptation and improvement of the quality system. The QA processes and documentation are being continuously revised and adapted to meet the changing necessities of the organisation.

Automate QS procedures application. Support the quality assurance activities according to the procedures defined in the QS.

Improve and increase the project controls. The availability of a set of indicators of the project trends improves the communication between the business managers and the managers or the running projects, allowing better and shared visibility of the project and reducing incomprehension and delays in performing corrective actions. Provide to the project (and business) managers the means to realise of the status of the project from the quality point of view.

Facilitate the exchange of information with customers and partners.

Automation of paper based internal communications aiming to the process improvement within the company organisation. Electronic publishing would increase the speed and breadth of information flow across the company to a degree unobtainable with paper.

Establish a historical record of the quality issues related to projects. This will result into a better effort estimation in future projects.

Project Organisation

The development of the overall PIE was split into two phases for the system implementation and experimentation respectively.

Phase I: System Implementation

The first phase was devoted to the definition and development of the new tools that support the needed infrastructure. To define the system, the first phase included the activities of *assessment* of the current situation of the GMV QS to provide findings and recommendations to facilitate improvements, and the *redesign* of the procedures of the QS to adapt them for automation when appropriate.

The QS assessment and system development followed a traditional life cycle. The system was set-up using state-of-the-art and mature technologies and commercially available tools, capable of meeting the necessary requirements of supporting classification, storage, search and retrieval, data collection or procedure automation. The ACQUASY was built upon following tools/technologies:

Intranet, providing the underlying communication infrastructure to connect the GMV staff to the Quality System. Internet, and in particular Intranet, arises as the technology that fits in the experiment requirements. An Intranet is an internal information system based on Internet technology. In particular the elements used to build the intranet included:

Apache web server.

Netscape browsers (version 4.5 or higher) to navigate through the information system.

Java development kit (JDK)

ApacheDBI

Database System to store and retrieve project data and quality procedures. ORACLE was selected as the relational database management system to manage the information about the QS and projects.

Office Software for retrieving, consulting and updating quality and project data. Microsoft Office 97 is used for this purpose as it is the standard office software imposed by the GMV QS.

Figure 1 shows the different elements that compose the ACQUASY System and the interactions between them.

A Pentium based PC with Linux operating system, is used as the information server in which the web and database servers were installed to dispatch users requests.

Each GMV engineer is assigned with one PC connected to the corporate LAN and MS-Office tools installed. The web browser Netscape 4.5 is installed in the user’s PC to enable access to the system.

Figure 2 shows how data are accessed through the use of the GMV computer infrastructure.

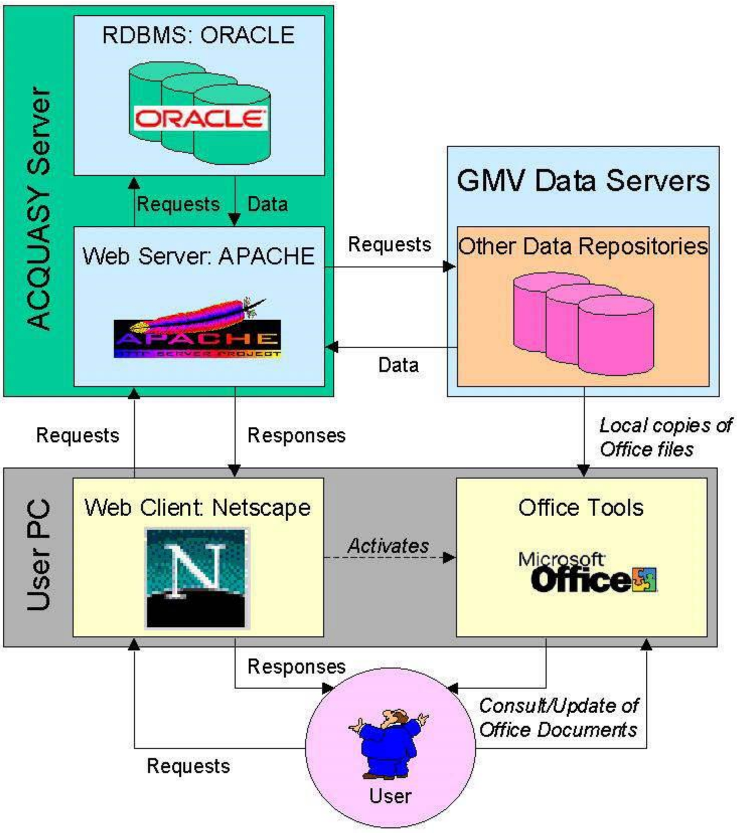


Figure 1 : ACQUASY Components

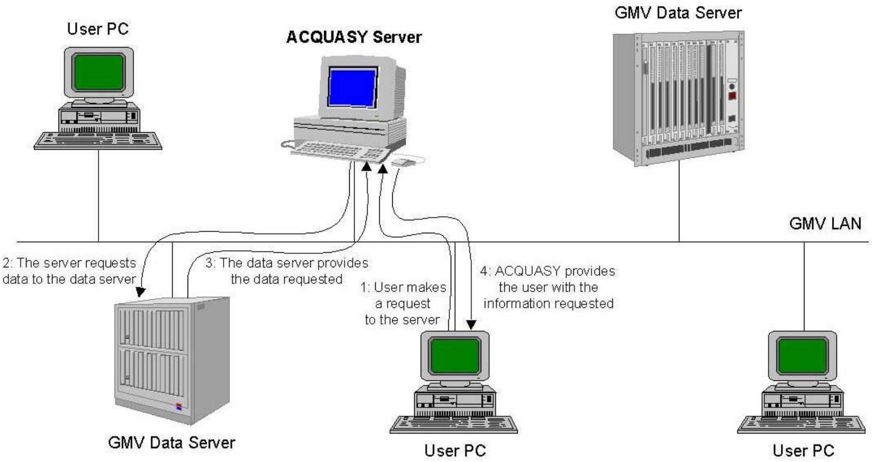

User PC
ACQUASY Server
GMV Data Server
GMV LAN
2: The server requests data to the data server
3: The data server provides the data requested
1: User makes a request to the server
4: ACQUASY provides the user with the information requested
GMV Data Server
User PC
User PC

Figure 2: Access to ACQUASY through the GMV Computer Infrastructure

Phase II: System Experimentation

The second phase of the project is being dedicated to the experimentation of the system and the measurements of results. The experimentation includes the activities of:

Selection of a baseline project, that is representative of the typical projects developed within GMV.

Training of the staff assigned to the baseline project to familiarise with the new system.

Experimentation of the new tool on the selected baseline project.

Collection of measurements on the benefits provided by the use of the new system. Currently, the system is being used by a pilot project. Project staffs as providing valuable feedback on the system itself and measurements are being defined and collected.

Results and Lessons Learned

The automated quality system has already been applied to a baseline project where between 30 and 40 developers are using it. The PIE is receiving feedback on the usability of the system and by the end of the year, it expects to have measured how well the system has been accepted and how it has influenced the product development.

There is already increased acceptance as a result of ease of access to the documents and ready guidance on how procedures should be applied. However, there are other paybacks for the automated system as well, particularly in facilitating configuration management and knowledge sharing across the company. Every new project will have an entry in the main page of the quality system, so all members of the development team have access to the same information in the project documents (and other types of items such as source code files, etc.). This eases the problem of configuration management across teams, which is proving particularly valuable.

Then, using the same Internet technology, all the documentation associated with a project can be made accessible to anyone in the company, so developers can search for other projects that have developed algorithms that can be reused, and new teams learn from past examples of other projects. Furthermore, experience in previous projects can be used for future estimations.

Instead of remaining local to projects, documentation is replicated to a central server in the company's main building and everyone has read-only access. Write access is password protected.

The feedback received from users up to now lead us to consider ACQUASY as a valuable tool for the GMV activities, so that its development will go beyond the PIE itself in order to improve it and provide it with the capabilities that eases the GMV development activities. As a matter of fact, users are providing valuable inputs in form of user requirements, for new capabilities of the system in the future.

The services provided by the system more appreciated by users are:

Access to the Quality System elements. Access to these elements has been enforced by the system.

Access to information of interest for different areas of the company. Users contributions to a common references spool, increased the accessibility to information by users.

After the first experiences by the baseline project, other projects requested their own entry in the system in order to exploit its capabilities. This saved time in preparing and maintaining configuration management procedures and tools.

The Company

Founded in 1984, GMV us a private Spanish company, fully independent from other national or international groups. Today, GMV´s turnover exceeds 12 MEUROs and the company employs over 200 people.

GMV activities cover the full system lifecycle, except for hardware manufacturing:

Technology and concept studies, including engineering services,

Definition, design, development, integration and maintenance of software systems.

Specification, design, procurement, integration, delivery and operations support for turnkey systems.

The major fields of activities of the company are:

Aerospace sector: GMV is heavily involved in European space programmes as a regular contractor to institutions and operators leading these initiatives. The company has had he opportunity of cooperating successfully with the most important European companies in the market, and has achieved the highest levels of competitiveness in the following areas:

Orbital Mechanics, being the only company recognised by the European Space Agency (ESA) as a Centre of Excellence in this discipline.

Satellite Control Centres Software, particularly within the Flight Dynamics subsystem.

Software for Satellite Data Processing, in particular those from Earth observation instruments.

Missing Analysis for space systems, contributing to the initial study and design activities of most ESA missions.

Global Navigation Satellite System (GNSS), geared towards the development of new systems as well as advanced applications.

Design and development of software for *Orbit and Attitude Control Systems*, or guidance, navigation and control, and space instruments.

Transport sector: Our offer within this market is oriented towards the improvement of operations efficiency and the increase of quality of service for all modes of transport, comprising:

Advanced air navigation and air traffic management systems.

Telematic systems for management and operations of transport means, both in airport facilities and maritime and terrestrial transport, including user information systesms.

Vehicle navigation and localisation systems, based upon advanced satellite radio-navigation means and mobile communications.

Training simulators for operations staff.

Simulators for design of transport systems and operations.

Systems for planning and management of companies logistics.

Defence sector: GMV is a supplier to the Spanish Armed Forces and participates in the research and development programmes of national as well as westerns institutions for defence and security (WEU, NATO), having the required security and quality certifications. We offer products and services mainly in the following areas:

Weapon systems simulators, aimed at being used for training as well as for development purposes.

Space systems applications for defence, such as observation satellites.

Embedded real-time software for military applications.

Satellite systems for precision navigation and localisation.

Training in advanced technologies especially in software engineering and simulation.

Telematic sector: When telecommunications and computer sciences have converged, GMV has capitalised on its traditional efforts in the development of total solutions for bussiness in areas such as:

Data processing centres for a variety of applications.

Decision aiding systems

Systems and tools for network management.

Distributed database systems.

Advanced graphics interfaces and multimedia.

Global solutions for connections to the information highways, designing and developing safe systems and providing value-added services based on INTERNET.

Industrial sector: The skills the we have acquired in the domains of systems modelling and simulation, development of control algorithms, software engineering for industrial environments and systems integration, allow GMV to provide customised solutions for the industrial market:

Manufacturing process management systems.

On-site integration of advanced industrial control applications.

Production processes monitoring and diagnosis.

The Author

Born in 1963, Oscar Tejedor is a Bch. in Computer Science since 1986. He joined GMV in 1988, where has participated in different projects and developments in several fields of the software engineering, including:

Artificial intelligence.

Definition and implementation of MMI.

Software verification and validation.

Internet developments.

Currently he is the Project Manager of the ACQUASY Project.

References

Implementation of ISO9000 in the Software Development (I & II, in Spanish). By Mirko Donié and Wolfgang Dette (IBM Germany). Published in Forum/Calidad 42/93.

Software Process Improvement via ISO9000? By Dirk Stelzer, Werner Mellis and Georg Herzwurm (University of Koeln, Germany). Published in Software Process - Improvement and Practice, Vol.2 197-210 (1996).

Web based Configuration Management System. By Gonzalo García (GMV, S.A.) (20 April 1998).

Pro-active or Reactive Quality? (in Spanish). By Giuseppe Satriani (European Software Institute, ESI), 12 March 1999.

Change Management System User's Guide. By the Scientific Project Department of the European Space Agency. October, 1997.

Review Item Discrepancy (RID's) System User's Guide. By the Scientific Project Department of the European Space Agency. October, 1997.

Document Management System WWW Interface User's Guide. By the Scientific Project Department of the European Space Agency. October, 1997.

Session 7
SPI and Assessments / Evaluations

Chairman

Tor Stalhane

Sintef, Trondheim, Norway

SPI Risk Assessment

by Allan Baktoft Jakobsen

Supporting SPI projects successfully requires detailed information about the project, and precise and quick feedback to the project. The TPR-PIF assessment method is designed to be used at a preliminary risk assessment meeting as a basis for further support. It has already been used by several companies participating in the European ESSI program.

The assessment will take 1-2 hours to complete and the aim is to obtain enough information about the project during the meeting to be able to list a few but precise bullets of feedback at the end. It should be emphasized that the assessment is not meant to give an objective or scientific measurement of the risks. The purpose is to facilitate a systematic discussion with the project in order to successfully address the risks and to reduce the resistance against improvements.

PIE project and Baseline project

The following model is the frame for the TPR-PIF method.

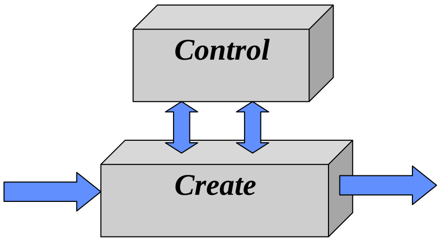


Figure: Create and control processes in a project.

In most projects, in particularly software projects, two fundamental processes can be recognized:

- The *Create* project, which eventually produces the final product of the project.
- The *Control* project, which controls and manages the *Create* project.

These two projects are obviously equally important.

For a PIE i.e. a Process Improvement Experiment (an ESSI term for a project working with software process improvements), we have

- *Create* project - *Baseline* project
- *Control* project - *Improvement* project

Thus, the changes initiated in the Improvement project should interact with the Baseline project, and hopefully lead to a better overall project.

Assessing the Baseline project using the TPR diagram

The TPR (Task-Process-Resources) model is developed by Allan Baktoft Jakobsen [IEEE Software Jan/Feb 1998] to provide a framework for capturing the various knowledge about a software project.

Examining the following generic process model, three different *points of view* can be identified:

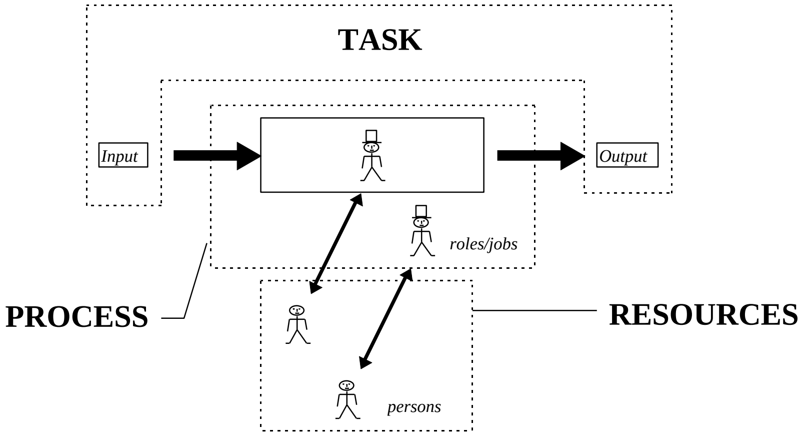


Figure: Three points of view in the generic project model.

These are:

The Task: This point of view focuses on the concrete *inputs* and *outputs* of the project. The ultimate output of the project is usually the product itself, but during the development a lot of sub-inputs and sub-outputs or internal deliveries are present. The Task often have primary interest of the management and sales people since the (external) inputs and outputs usually involves interacting with customers. Questions about the task are usually *What-questions*.

The Process. This point of view focuses on the *transformations* of inputs to outputs. If the transformations can be generalized and described in an abstract form independent of the concrete input and output in order to be reused we have the concept of a *process*. If the task focused on the starting point A and goal B of the projects, the process is about the map of the way from A to B. Questions about the process are usually *How-questions*.

The *Resources*: This point of view sees the project in terms of the *people* of flesh and blood who play the roles defined in the process. Questions about the resources are usually *Who-questions*.

A baseline project can be systematically examined in the frame of these three points of view.

The TPR assessment process

The TPR assessment process is the following:

Input: Documents and people from the project.
Transformation: Questions in the frame of the TPR model. Evaluations of answers.
Output:

- Timeline of the project.
- Organization diagram.
- TPR-diagram.

Examples of questions are listed below. There are 9 groups of questions corresponding to the 9 areas of the TPR diagram. The list is not complete. It depends on the assessors feelings and knowledge of software development to ask the optimal questions.

When the questions for a given area are asked and discussed, the assessor makes for himself a quick decision regarding the following question:

- Is there anything in this area that is an obvious risk and how critical is it?
- If there are many critical risks, a mark should be plotted close to the center of the TPR kiviat diagram (Short radius.)
- If there are no major risk or they are under control, a mark should be plotted close to the periphery of the kiviat diagram (Large radius.)

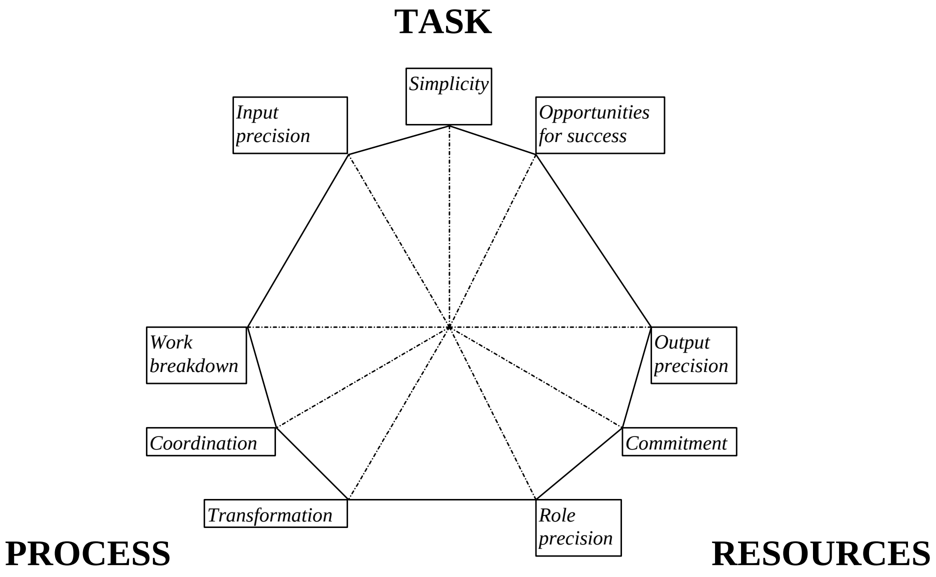


Figure: TPR diagram for assessing the baseline project.

When the 9 marks have been plotted into the TPR diagram, they are connected to a polygon. A discussion of the findings can now begin:

- If the polygon is smooth, round and large there are no critical risks.
- If the polygon has bumps, potential risk areas are visualized.

Task questions – General

- Describe the products of the project.
- Describe the vision and goals.

- Describe the customers and the market.

Task questions – Input Precision

- Describe the requirements to the project in terms of: Documentation, level of details, completion, consistence, and correctness.
- What is your relation to the customers?
- What is your company's background in the technical domain?
- Describe the internal documentation in terms of level of details, completion, consistence, and correctness.

Task questions – Simplicity

- What is the complexity of the product?
- What is the size of the product?
- What technology is used in the product? How well-known is it? How advanced is it?

Task questions – Opportunities for success

- What is the importance of the product compared to the other products from the company?
- If the product is successful on the market how significant will it be to the company?
- If the product fails what does it then mean?
- How prestigious is the project?

Process questions – General

- Describe the mission of the project.
- Draw a *time line* of the project.

Process questions – Work break-down

- How is the work break-down in the project?
- Is it documented?
- How detailed is it?

Process questions – Coordination

- How is the work in the project coordinated? Are there any defined processes?
- How does the organization support the coordination?
- How is the planning?
- Is there a documented project plan? (Let us see it, please!)

Process questions - Transformation

- How do you intent to go from A: The *requirements* to B: The *product*
- How is it ensured that the goals of the project are reached?
- Are there any defined processes for the transformation?
- How does the organization support?

Resource questions – General

- Draw an organization diagram of the project. Don’t forget the names of the persons involved.

Resource questions – Role precision

- Are the persons involved aware of their specific role in the project? (as analyst, designer, programmer, tester, etc.)

- What is the process competence/skills of the various persons involved? (Do they know how to analyze, structure, plan, design, produce, check and test?)
- What is the social competence of the various persons involved?
- How precise is the match between process and people in the above areas?

Resource questions - Motivation

- What is the motivation of the persons involved?
- Who is taking the initiative in the project?
- Does anyone come up with new ideas?
- Does people work on overtime?
- How is the work atmosphere?
- How is the work environment?

Resource questions – Output precision

- Are the persons involved delivering the output they are supposed to? At the right time?
- What is the technical competence of the persons involved?
- How is the performance of the people?

Overall triangle

The faces of the TPR triangle, that is, the relations between T-P, T-R, and R-P, are sometimes called the dimensions of Management, Leadership, and Dedication, respectively. How does the company value these dimensions when decisions are to be made?

Assessing the Improvement project using the PIF diagram

The PIF (Process Improvement Footprint) model was proposed by Chuck Myers and Suzanne Garcia at the E-SEPG 98 conference in London and slightly restructured by Allan Baktoft Jakobsen.

Improving processes and changing in general has little to do with technology but a lot to do with human beings. So the PIF diagram is about them. People in relation to change can be assessed from three different points of view: As individuals, as groups, and as organizations. This is the essence of the diagram.

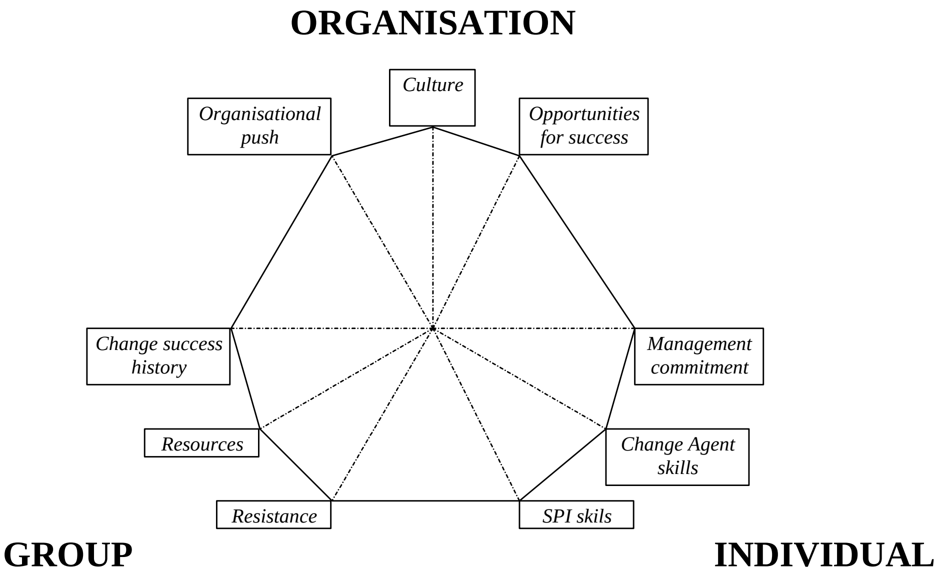


Figure: PIF diagram for assessing the improvement project.

The PIF assessment process

The PIF assessment process is the following:

Input: Information from the people from the project.
Transformation: Questions in the frame of the PIF diagram. Evaluations of answers.
Output:

- PIF-diagram.

When the 9 marks have been plotted into the PIF diagram, they are connected to a polygon. A discussion of the findings can now begin. Again:

- If the polygon is smooth, round and large there are no critical risks.
- If the polygon has bumps, potential risk areas are visualized.

Below the areas of questions are listed.

Questions on individuals – Management commitment

- How is the management commitment? Does the top management pay any interest? Do they show up at your meetings?
- Who is sponsor for the improvement project? Who is paying?

Questions on individuals – Change Agent skills

- Is there a champion among? (i.e. a person with the personal authority and charisma to drive the change.)

Questions on individuals – Technical skills in SPI

- Which SPI skills are present?
- Has the SPI manager carried out SPI before?

- What knowledge on *best practiced* is present?
- Does the SPI manager attend SPI networks or conferences? Do they read papers or books?

Questions on groups – Resistance

- What's the level of resistance among the people?
- If the resistance visible or hidden?
- How old are the people involved?

Questions on groups – Resources

- What is the resource situation for the improvement project compared to the general situation in the company?

Questions on groups – Change success history

- Does the company have any recent success histories about improving and changing the processes?

Questions on the organization – Organizational push

- What is the aggregated attitude to improvements from the organization, that is the top managers, the middle managers, the developers, the culture, and the business areas operated?

Questions on the organization – Culture

- How is the company culture in relation to quality, improvements, and change?
- Who generates ideas for improvements?
- Who take responsibility for improvements?
- Who are the drivers of change?

Questions on the organization – Opportunities for success

- If the improvement activities are successful, what will the consequences be?
- If the improvement activities fail, what will the consequences be?

Overall triangle

Are the decisions driven by: The individuals, the groups, or the organization? What level of capability maturity do you suspect of the company?

Final assessment: TPR and PIF results combined

After the meeting with the SPI project the assessors discuss the results of the TPR and PIF diagrams.

An ordinary *SWOT* (Strengths, Weaknesses, Opportunities, Threats) list with 5-6 bullets is produced and mailed to the project.

Experiences with TPR-PIF

Anyone who has been involved in software process improvements know how difficult it can be. Many times the battle of convincing/persuading the project managers and the developers to go on is not really won. Pressure from top management cannot avoid people resisting the proposed changes.

Resisting change is a key area of SPI work and although the reasons for this phenomenon is fairly well understood, it is seldom systematically counter-measured. In fact, it's all about insecurity and fear of loosing control - in other words, a very human reaction.

The TPR-PIF method has been successfully used by DELTA in Copenhagen, by FZI in Karlsruhe, and on Iceland. These companies are all *EspiNodes* in the ESSI program. The number of assessed projects (*PIE's*) is currently about 20. The method is still being refined.

As mentioned earlier, the assessment is not an attempt to measure the risks in any quantitative way. The main purpose as we have seen it in practice is that it opens up for a qualitative yet highly *structured* discussion of the risks of carrying out improvements. The awareness and overview of the potential risks have a double purpose. First, to reduce the total risk of baseline project failure and second, to reduce the resistance against the improvement project due to insecurity and lack of knowledge

In the interviews we have noticed the effect of discussing the problems of both the baseline project and the improvement project from the various points of view. The frame provided by the TPR-PIF method ensures that most of the important topics are covered. Moreover, the final summery using few very simple kiviat diagrams is an excellent way of sharing the overview of the current situation. If done properly, key insight can be gained here.

Traditionally, risk analysis is hard. Project managers say that this is what they are doing all the time. Developers say that there are two kinds of risks: The ones you can do something about and the ones that are simply out of your sphere of power and control. The first ones occupy most of your time.

Thus, in a hectic software project, all the things to do and to be aware of soon seem overwhelmingly many. Proposing process improvements on top of all that is bound to provoke resistance. The TPR-PIF method helps by bringing in the overview that is blurring the intellectual control of the project. This means reducing the insecurity in the project by increasing the knowledge of the real problems.

References:

1. Allan Baktoft Jakobsen: Bottom-Up Process Improvement Tricks, IEEE Software, Jan./Feb. 1998, pp. 64-68.
2. Allan Baktoft Jakobsen: Over My Dead Body, The SPIDER Kourier, October 1999.
3. Chuck Myers, Suzanne Garcia: SEPG SkillShop: Building Your Process Improvement toolkit. E-SEPG 1998.
4. Gerald Weinberg: Quality Software Management Vol. 3: Congruent Action. Dorset House 1994.
5. Gerald Weinberg: Quality Software Management Vol. 4: Anticipating Change. Dorset House 1997.
6. Watts S. Humphrey: Managing Technical People. Addison-Wesley 1997.
7. Susanne Kelly: The Complexity Advantage: How the Science of Complexity Can Help Your Business Achieve Peak Performance. McGraw-Hill 1999.

Allan Baktoft Jakobsen has an educational background in mathematics and physics from Copenhagen University and Dartmouth College. He has been working in the software business with maintenance, design, coding, test, quality assurance, and as an SEPG member. He has recently founded his own company, MindMate, counseling best software practices and process improvements. Contact Jakobsen at baktoft@mindmate.dk.

Rethinking the Concept of Software Process Assessment

Tore Dybå, M.Sc.
Nils Brede Moe, M.Sc.
SINTEF, Trondheim, Norway

Introduction

Much of the discussions on software process improvement (SPI) during the 1990s have focused on software process assessment and "best practice" models such as the Capability Maturity Model (CMM) for software [11], and ISO/IEC 15504 (SPICE) [5].

In this paper we present a critique of the global "best practice" approach to software process assessment and improvement, focusing on the necessity to explore the contingencies of individual software organisations. Furthermore, we present some of our experiences in using tailor made assessments based on a participative approach to focus software process improvement activities in Norwegian software companies.

The participative approach to software process assessment is part of the methodological basis used in a major Norwegian SPI program called SPIQ (Software Process Improvement for better Quality). The objective of SPIQ is to increase the competitiveness and profitability of Norwegian IT-industry through a systematic and continuos approach to process improvement.

The goal of SPIQ is twofold: (1) to establish an environment for process improvement in software companies associated with SPIQ, and (2) to transfer and diffuse the knowledge gained to the remaining IT-industry in Norway through training, seminars, and conferences.

The rest of this paper is organised into five sections. In the first section we discuss the role of assessment in SPI. Next, we present a participative approach to software process assessment, and our experiences with using this approach is then exemplified with two case studies. Finally we summarise our experiences in terms of lessons learned, and lastly we make some concluding remarks.

The Role of Assessment in SPI

An increasingly popular way of starting a SPI program is to do an assessment in order to determine the state of the organisation's current software processes, to determine high-priority issues, and to obtain organisational commitment for SPI.

Why Perform Assessments?

Not all software companies are equally skilled at identifying the causes of their problems or to identify the most rewarding opportunities for future competition. Without a preliminary problem analysis, "solutions" are seldom effective; on the contrary, they are often irrelevant to the underlying causes of the symptoms that are being treated and only add more noise to the system. Consequently, it is of utmost importance that complex problems in software development must be thoroughly understood before a solution is attempted.

In many cases, process assessment can help software organisations improve themselves by identifying their critical problems and establishing improvement priorities before attempting a solution [6]. Therefore, the main reasons to perform a software process assessment is [13]:

1. To understand and determine the organisation's current software engineering practices, and to learn how the organisation works.
2. To identify strengths, major weaknesses and key areas for software process improvement.
3. To facilitate the initiation of process improvement activities, and enrol opinion leaders in the change process.
4. To provide a framework for process improvement actions.
5. To help obtain sponsorship and support for action through following a participative approach to the assessment.

As described later, the last point – a participative approach – is crucial for a successful software process assessment.

Ways of assessing software processes

There are three ways in which a software organisation can make an assessment of its development practices:

1. Benchmark against other organisations.
2. Benchmark against "best practice" models.
3. Assessment guided by the individual goals and needs of the organisation.

The first way of doing an assessment is a traditional benchmark exercise used to gain an outside perspective on practices and to borrow or "steal" ideas from best-in-class companies. This type of benchmarking is "an ongoing investigation and learning experience that ensures that best practices are uncovered, analysed, adopted, and implemented." [3]

Hence, benchmarking is a time-consuming and disciplined process that involves (1) a thorough search to identify best practice companies, (2) a careful study of one's own practices and performance, (3) systematic site visits and interviews, (4) analysis of results, (5) development of recommendations, and finally, and most importantly, (6) implementation.

The second way of performing an assessment is to benchmark the company against one or more of the "best practice" models on the market. Over the years, several assessment models have emerged in the software industry, and there is a range of possible assessment models that one can choose from. In addition to the CMM for software and ISO/IEC 15504, further examples of such models are ISO 9001 (including 9000-3), TickIT, the European Quality Award, Bootstrap, Trillium, and ISO/IEC 12207.

The models focus on different aspects of the software processes and the organisation, and they are all associated with specific strengths and weaknesses. However, they share a common set of problems, which mainly has to do with the fact that they are artificially derived and based on idealised lists of unvalidated practices. Besides, they are associated with both statistical and methodological problems [2].

Furthermore, most of these models also emphasise an improvement approach based on statistical process control (SPC), which is a highly questionable approach for the majority of software companies [10].

The third way of performing an assessment is with a participative approach tailored to the individual needs of the company. This approach is less time-consuming than the traditional benchmark approach, and it is clearly more relevant and valid than the model-based approach.

During the 1990s, "software process assessment" has become synonymous with the model-based approach. In our view it is time to rethink this conception of software process assessment, and to proceed to a tailor made and participative approach focusing on what is unique to each company and how this uniqueness can be exploited to gain competitive advantage.

A Participative Approach to Software Process Assessment

The objective of our approach to software process assessment is to focus on the necessity of participation for SPI to take place. Basically, there are three reasons for this: (1) Developers and managers alike must accept the data from the assessment as valid, (2) they must accept responsibility for the problems identified, and (3) they must start solving their problems.

General principles for performing model-based software process assessment are given in [6, 7, 11, 13] and specific guidelines for performing CMM-based assessments are given in [4] and for ISO/IEC 15504 conformant assessments in [5].

Participative Assessment Process

We adopted a general approach for organisational assessment and specialised it to the domain of software development. The process involved researchers and practitioner acting together in a participative approach to diagnose problems in software development using the basic principles of survey feedback (see e.g. [1, 8]) which is a specialised form of action research.

The assessment process is an adaptation of the evaluation model developed by Van De Ven and Ferry [12] and consists of six steps, as shown in Figure 1, and described below:

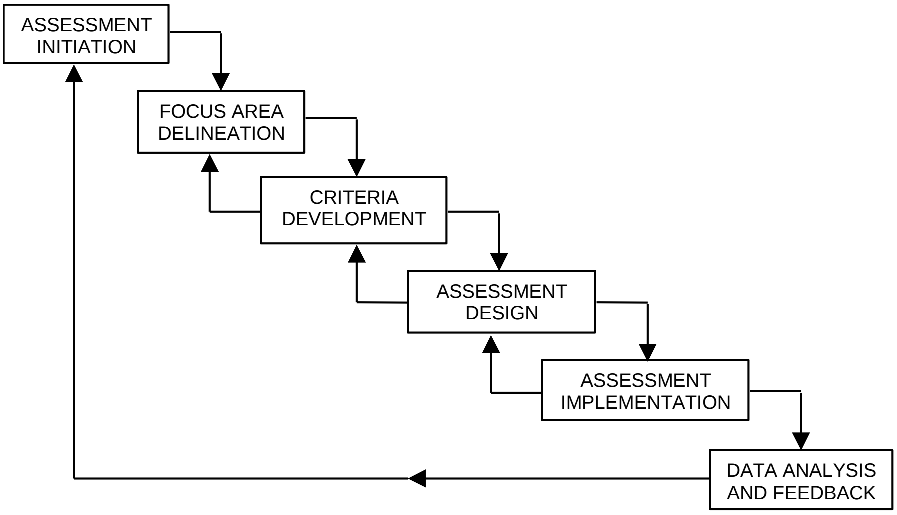


Figure TDNBM.1: The Assessment Process.

In the first step, *assessment initiation*, the insiders and outsiders of the organisation should clarify their respective roles and the objectives of the assessment by answering the following questions:

1. What are the purposes of performing software process assessment?
2. Who are the users of the assessment, and how will the results be used?
3. What is the scope of the assessment in terms of organisational units and issues?
4. To what extent is there a commitment to using scientific methods (e.g psychometric principles) to design and implement the assessment?
5. Who should conduct the assessment, and what resources are available?

It is important that due considerations are taken in answering these questions, since they are crucial for determining whether an assessment is relevant in the first place, and for tailoring the process and content of the assessment to the specific needs of the organisation.

The second step, *focus area delineation*, is an exploration of the overall issues identified for the assessment in step one. In our experience, most companies do not have a shared understanding of their specific goals. A conscious analysis of commonly used high-level performance goals and focus areas in standards and reference models can, therefore, be useful as a starting point for group discussions in this step. Examples of such focus areas are software processes (e.g. customer-supplier, engineering, support, management and organisation), competitive priorities (e.g. price, quality, flexibility, and service), organisational learning (learning from past experiences, learning from others, and current SPI practices), and perceived factors of success.

In the third step, *criteria development*, multiple operational criteria are developed for each of the high-level goals. The process of criteria development requires that practitioners select and define concrete characteristics that are to be measured and used as indicators of goal attainment. A decision also has to be taken regarding the use of aggregate or composite measures.

To operationalise the criteria, one question is defined for each characteristic such that the theoretical abstractions can be closely related with everyday work situations. We adopted the format used in the European Software Institute's 1995 Software Excellence Survey, such that two subjective rating scales accompany each question: one to rate the current strength or practice level and one to rate the future importance (see Figure TDNBM.2).

Furthermore, to help the companies, we developed a standard questionnaire that could be used as a starting point for internal discussions and for the development of tailor-made questionnaires. The standard questionnaire is based on our experiences of performing process assessments in six companies during the SPIQ pre-project phase, and includes four sections. The first section, on competitive priorities, is adapted from the aforementioned ESI survey. The second section is adapted from the software process areas in the emerging ISO/IEC 15504 standard. The third section concerns SPI processes and learning from past experiences and the experiences of others. Finally, the fourth section is concerned about finding the most important factors enabling SPI success in the organisation.

Current strength						**Future Importance**				
1	**2**	**3**	**4**	**5**	**DELIVERY**	**1**	**2**	**3**	**4**	**5**
□	□	□	□	□	*Ability to deliver on schedule*	□	□	□	□	□

Figure TDNBM.2: Typical question format from the questionnaire.

The issues pertinent to step four, *assessment design*, relate to where the assessment will be conducted (organisational units), the role of the insiders and outsiders, the time horizon of the assessment, the unit of analysis, as well as deciding what the sample would be, how the data will be collected, how aggregate concepts will be measured, and how the data will be analysed.

When aggregate or composite measures are used, one should be careful about deciding the corresponding requirements of psychometric properties (see e.g. [9]). That is, unless the composite scales in the questionnaires are constructed and evaluated along the lines associated with psychometric tests, they may produce assessment results that are seriously misleading.

It is important, however, to note that the more rigorous the assessment design becomes, the greater the time, costs, and other resources expended on the assessment are likely to be. Therefore, one should ask the question at every decision point whether the benefits that result from a more sophisticated design to ensure accuracy, confidence, generalisability, and so on, are worth the investment of more resources.

In step five, *assessment implementation*, the assessment is implemented according to the procedure decided upon in the previous step. The main considerations during this step are completeness and honesty in data collection procedures and the recording of unanticipated events that may influence the assessment results.

The major concerns during step six, *data analysis and feedback*, are to provide opportunities for respondents to participate in analysing, interpreting and learning from the results of the assessment. And, furthermore, to identify concrete areas for improvement. There are many ways in which this could be done. We have relied upon half-day workshops in which preliminary findings on initial questions and problems are presented verbally, in writing, and with illustrations.

These workshops begin with a review of the objectives of the assessment, the focus areas and the design and implementation of the assessment. Findings regarding the scores on current strengths and future importance are presented in terms of a gap analysis. Normally, the participants raise a multitude of questions and issues when the findings are presented, and they take part in group discussions and reflections as they review and evaluate the preliminary findings. Some of the questions can be clarified and answered directly with the data at hand, other questions can be answered by reanalysing the data, and finally some issues are raised which cannot be resolved with the current assessment data. In the last case, a decision has to be taken regarding further data collection.

Typically, we use scatter plots, bar charts and histograms to illustrate preliminary findings from an assessment in order to highlight gaps between current levels of practices and future importance and the dispersion of responses both within and between groups.

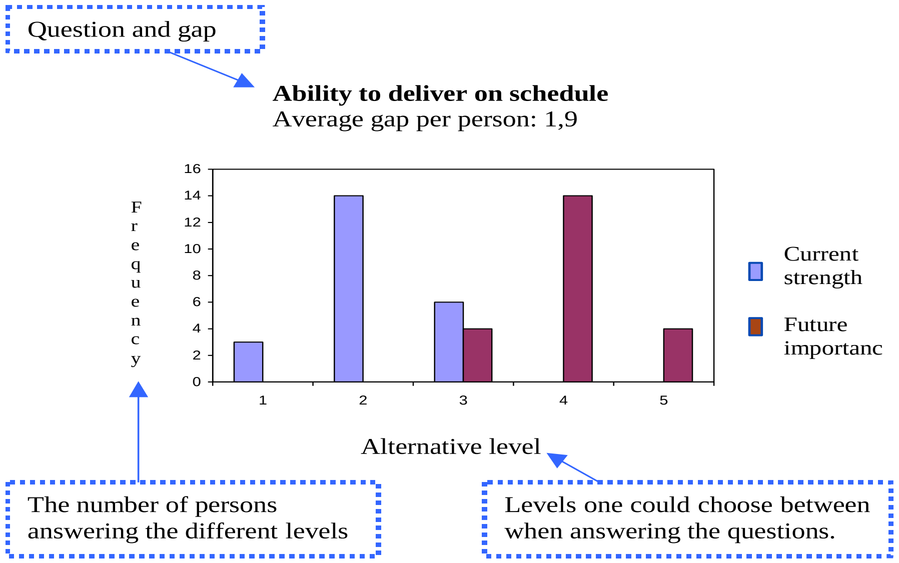


Figure TDNBM.3: Illustration of preliminary findings.

Figure TDNBM.3 shows an example of an illustration that was used in a feedback session presenting preliminary findings from an assessment. Some of the information was presented verbally. The extra information for this graph was “100% of the respondents have a gap that is larger than or equal to one”.

Model-based approach versus participative approach to assessment

In summary, our approach to software process assessment is based on a structured process emphasising participation in each and every step. See Table TDNBM.1 for a comparison of the model-based approach and the participative approach to software process assessment.

Feature	Model-based Approach	Participative Approach
Focus areas and criteria from	"Best practices" according to the reference model.	Tailor made to the needs of the organisation.
Data collected from	Selected group of managers and representatives from specific projects.	Everyone in the organisation or department.
Data reported to	Sponsor (top management and department managers).	Everyone who participated (including management).
Role of researcher or consultant	Administration of questionnaires, documenting findings and recommendations.	Obtain agreement on assessment approach, joint design and administration of questionnaire, design of workshops.
Action planning done by	Top management.	Teams at all levels.
Probable extent of change and SPI	Low.	High.

Table TDNBM.1: Two approaches to software process assessment.

Case Studies

In this section, we take a look at how two companies made significantly different implementations of the participative approach to software process assessments and in the next section, we present the key lessons learned from these cases.

Company Y

Y is nearly 15 years old, and has grown to become one of the leading consulting companies in Scandinavia. Current customers include large industrial enterprises, government agencies and international organisations. They focus on using an iterative and incremental development method. The company has a flat and open organisation, with a process-oriented structure, and employs about 140 persons. Over 90% of these hold a MSc/MBA.

In the first step (*assessment initiation*) members from the SPIQ project team had an opening meeting with the manager of technology. This meeting resulted in formulation of two objectives for the assessment:

1. Get a survey of today's process status.
2. Get an "outsiders" view on the company to suggest areas for improvement.

The assessment was mainly focusing on project managers and department leaders. All the questions in the assessment were related to software development. The scope of the assessment in terms of organisational units and issues were all the process-areas of the company. The assessment was decided to be conducted by the SPIQ-team and the manager of technology.

In steps two and three (*focus area delineation* and *criteria development*), Y used the standard questionnaire as a starting point for internal discussions, and for the development of tailor-made questionnaires. They did not change anything, which was a bit surprising. The purpose of doing this was the wish for external impulses. No aggregate or composite questions were used; the focus was only on single questions.

In step four (*assessment design*), the date of the assessment was determined, and it was decided that one of the researchers from SPIQ should hold a presentation for the managers in the company. This was to be followed by the assessment. The presentation was an introduction to SPI with the focus on general improvement. The purpose of this was to describe the questionnaire and its purpose, and also have a quick walkthrough of the questions.

After a short period of planning, all was set for stage five (*assessment implementation*). After the presentation, the participants (10 persons) from Y filled out the questionnaires in the meeting-room. This gave them the opportunity to discuss and agree upon the interpretation of unclear questions in the questionnaire. The participators in the assessment only answered for the unit and those levels that were part of their own area of responsibility. All the information was treated confidentially. Filling out the questionnaire took about 30 minutes.

In the final step (*data analysis and feedback*), most of the respondents participated in the analyses and interpretation of the preliminary result. A half-day workshop was set up for this event. The most important results were then presented. The participants raised a lot of questions and issues , and they started a big discussion as they reviewed and evaluated the preliminary findings. Some of the questions were clarified and answered directly, others were answered by reanalysing the data.

The discussion ended in a priority list of four key areas. These were: delivery (time-schedule and budget), customer supplier-relationship, testing and configuration management and risk control. Some of these results were not expected, while others were obvious to the participants.

The next step for company Y will be an internal discussion of the results, and then figure out the necessary SPI actions. However, they are first going to co-ordinate this work with the work going on in a parallel project.

The result of the last section of the questionnaire was also of great interest. This section is divided into tree sub-sections, and is concerned about finding the most important factors enabling SPI success in the organisation. The most important arguments in favour of SPI in the company Y were: Continuous adjustment to external conditions, job satisfaction, and vital importance for future competitiveness. The most important arguments against SPI were: Increasing workload (resource demanding), SPI suppresses the creativity and the sense of responsibility, and it moves focus away from the project.

The most important factors for ensuring successful process improvement in the company were: Management involvement, motivation/employee participation, and well-defined and simple routines.

Company X

X is one of the leading companies in their field. Their products are a combination of software (embedded software or firmware) and hardware. In addition to corporate offices and manufacturing facilities in Norway, X has significant marketing, sales and support operations in the USA, Europe, and the Far East. The company employs about 550 persons in Norway, of which the firmware division employs 30 persons.

During the first step (*assessment initiation*), members from the SPIQ project team and company X had an opening meeting where the objectives of the assessment were set. X wanted to get a survey of today's process status and potential for improvements. After identifying some key areas of improvement, the intention was to make a plan for SPI-actions. The assessment was focusing on project managers, customers and co-workers in two firmware departments and one hardware department. These groups were divided into tree subgroups: Managers, customers and developers. All departments in company X were represented. The assessment was decided to be conducted by the SPIQ-team and the people responsible for the assessment at company X.

In steps two and three (*focus area delineation* and *criteria development*), X used the standard questionnaire as a starting point for internal discussions, and for the development of a tailor-made questionnaire. A committee was put together for evaluating the questions. In this committee there were two designers, four managers, two customers, and one from the quality department.

After two meetings the number of questions were doubled. None of the original questions were removed. We recommended X to reduce the number of questions, and to do some minor changes in the wording of items, in order to be more precise in the question text. Some rewording was subsequently done. However, the number of questions was not reduced. No aggregate or composite questions were used; the focus was only on analysing answers to single questions.

During step four (*assessment design*), it was decided that the assessment co-ordinator in the company should hold a presentation regarding the assessment's role in improving X's software processes. In step five (*assessment implementation*), the presentation was hold. Most of the participants in the assessment were present at this meeting. After the presentation they had 30 minutes to fill out the questionnaire. This was too little time, however, so almost everyone had to deliver the questionnaire later. The leader of the assessment personally visited those who did not show up for the presentation helping them to complete the questionnaire. 32 employees from company X participated, however, four did not deliver the forms. The participators in the assessment only answered for the unit and those levels that were part of their own area of responsibility. Most of the respondents participated in the analyses and interpretation of the presented result in step six (*data analysis and feedback*). A half-day workshop was set up for this event. The most important results were then presented. The problem with this session was the lack of discussion. Although this complicated the process, the session ended with a priority list of four key areas. These were: Flexibility vs. stability, long-term quality, teamwork, and learning from past experiences.

The next step for company X will be an internal discussion of these results, and to start a process to suggest alternative SPI actions to start with. This job is a bit difficult at the moment because company X is in the middle of a rearrangement.

The result of the last section of the questionnaire was also of great interest. The most important arguments in favour of SPI in the firmware-group were: Quality, motivation/employee participation, and job satisfaction. The most important arguments against SPI were: "This only creates new procedures and rules" and "a waste of recourses (bad priority)."

The most important factors for ensuring successful process improvement in the firmware-group were: Motivation, developer/designer in focus, and management involvement

Discussion of the cases

These cases are from two quite different companies; Y, a pure software company and X, a combined software and hardware company with their own production. They both had the same method to follow, but the accomplishment was quite different in a lot of areas. The objectives of the assessment was much the same.

Company Y did not work on the questionnaire template, and let the researchers perform the assessment. The questionnaire was therefore not as tailor-made as one would expect. The reason for this was, as explained before, the wish for only external input to the assessment. If this way of conducting the assessment is successful or not is too early to conclude.

On the other hand, company X did a lot of adjustments and therefore developed a highly tailor-made questionnaire. The problem with this case was the lack of involvement from the researcher's side. To many questions were produced without removing any from the template. Too many questions were too similar, and there were problems interpreting some of them.

With this situation in mind, one could expect that there would be a great discussion on the result from the tailor-made questionnaire, and less discussion on the result from the standard questionnaire. It was a big surprise that the opposite occurred. There could be a lot of reasons for this: At company X over 20 persons participated in the discussion, at Y there were only 10 persons. Also, the participants at X were a mixture of managers and developers, and there is a possibility that this prevented people from speaking out.

Another distinction between the two companies is the composition of the groups that participated in the assessment. In company Y, the group was homogenous (only process managers), but in company X there were three different groups. In this kind of assessment, the results are more interesting if there is a large group answering the questions, and if they come from different parts of the companies. This was the case at company X.

Comparing data from different groups and between different members of the same group gave interesting results. For example did Project manager have the opinion that the level in "Current strength" (topic: "making fast design changes") was low and that one should improve this area significantly. The developers had the opposite opinion. They meant the level today was too high, and wanted to decrease it. People from the customers group thought the level was OK.

The results from the discussions had very little in common. The results from the fourth section of the questionnaire had more in common. Under the category “The most important arguments in favour of SPI”, the results tell us that both companies think that SPI activities will improve quality and make the employees more satisfied. SPI activities seem like a necessary thing to do if you want to achieve success.

In the category “The most important arguments against SPI”, the companies had the same opinion on SPI-work increasing the workload and as a source of new procedures and rules, which will cost a lot of resources and move the focus away from the projects. Comparing the results from these categories is interesting, because they first argue that SPI is necessary for the company to survive, but there is a lot of negative work to be done doing this. Maybe SPI has a problem with the association of “quality control”!

Under “The most important factors for ensuring successful process improvement”, the companies had two factors in common: Employee participation and management involvement.

During the presentation of the data, there were a lot of “expected” conclusions, but also some the company had never thought about. The conclusions they had expected had, however, never been externalised before. This happened for the first time as a result of the assessments.

Lessons Learned

1. *Assessment initiation*

   - By involving more than one group in the assessment, there exist possibilities for multiple views and interest in the discussion and analyse phase.

2. *Focus area delineation and criteria development*

   - A team should be put together in the company performing the assessment in order to construct a tailor-made questionnaire.
   - The time needed to complete the questionnaire should not exceed 30 minutes (about 60 questions). If there is a problem with not covering all the areas wanted in the initiation, the company should conduct more than one assessment.
   - There should be a close co-operation between the outsiders and the insiders. Companies have problem to distinguish between important and less important questions. They may also have a problem with noticing if two or more questions express the same concept. In such cases, people from the outside will be able to help.
   - It is very important for the outsiders to get to know the company well enough to be able to give useful advice. To achieve this, it is critical that they work closely together with the people in the company performing the assessment.

3. *Assessment design*

   - Hold a presentation for the persons participating in the assessment. This presentation should be both motivating for the assessment, and secure that everybody has the same understanding of the questions and the goals of the assessment.

- During the assessment, there will always be discussions regarding the interpretation of questions. It is therefore advantageous to let the participants conduct the assessment at the same time.
- All information should be treated as confidential. If not, there will always be someone not speaking from the heart.

4. *Assessment implementation*
   - Do not wait too long before performing the feedback-session. This may lead to loss of SPI focus in the department/company.
   - If definitions of questions are discussed during implementation, it will be wise to document the conclusions of these discussions. During the analysis and feedback session, it is highly possible that these questions will be discussed again, and the participants will not remember how they interpreted these.
5. *Data analysis and feedback*
   - Do not leave this session without identifying concrete areas for improvement. These areas will be input to the next phase of improvement – action planning.
   - To have a useful discussion in this session, make sure the group participating is not too large or too small. If there is a big group, the assessment leader in the company should prepare the data to suggest some key areas before the half-day workshop. The ideal size of the group is 8 – 14 people.

Maybe the most valuable lesson learned was the need for the assessment to be closely aligned with and tailored to the company’s overall strategy. Without this it will be hard to get acceptance by the managers, and this will lead to fewer resources and low priority. It is also necessary with a tight time-schedule. Waiting too long between the steps will only lower the interest and motivation.

Conclusions

In this paper we have described a participative approach to software process assessment, experiences from two divergent implementations and the lessons learned. In summary, SPI efforts should be tailored to the goals and needs of the individual organisation, not benchmarked against a “synthetic” model of so called “best practices”. Furthermore, researchers and practitioners should work closely together in a mutually accepted setting of inquiry, action and learning to solve the problems at hand and to explore possibilities of competitive advantage through improved development processes.

References

[1] Baumgartel, H. Using Employee Questionnaire Results for Improving Organizations: The Survey ‘Feedback’ Experiment, in *Kansas Business Review*, Vol. 12, pp. 2-6, December, 1959.

[2] Bollinger, T. and McGowan, C. A Critical Look at Software Capability Evaluations, in *IEEE Software*, pp. 25-41, July, 1991.

[3] Camp, R.C. *Benchmarking: The Search for Industry Best Practices that Lead to Superior Performance*, Milwaukee, Wis.: ASQC Quality Press, 1989.

[4] Dunaway, D.K. and Masters, S. “CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description,” Technical Report, Software Engineering Institute, CMU/SEI-96-TR-007, 1996.

[5] El Emam, K., Drouin, J.-N. and Melo, W. (eds.) *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, Los Alamitos, California: IEEE Computer Society Press, 1998.

[6] Humphrey, W.S. *Managing the Software Process*. Reading, Massachusetts: Addison-Wesley, 1989.

[7] Humphrey, W.S. *Managing Technical People: Innovation, Teamwork, and the Software Process*. Reading, Massachusetts: Addison-Wesley, 1997.

[8] Neff, F.W. Survey Research: A Tool for Problem Diagnosis and Improvement in Organizations, in A.W. Gouldner and S.M. Miller (eds.) *Applied Sociology*, pp. 23-38, New York: Free Press, 1966.

[9] Nunnally, J.C. and Bernstein, I.A. *Psychometric Theory*, 3rd edition, New York: McGraw-Hill, 1994.

[10] Ould, M.A. CMM and ISO 9001, in *Software Process – Improvement and Practice*, Vol. 2, pp. 281-289, 1996.

[11] Paulk, M.C., Weber, C.V., Curtis, B. and Chrissis, M.B. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, Massachusetts: Addison-Wesley, 1995.

[12] Van de Ven, A.H. and Ferry, D.L. *Measuring and Assessing Organizations*, New York: John Wiley & Sons, 1980.

[13] Zahran, S. *Software Process Improvement: Practical Guidelines for Business Success*, Harlow, England: Addison-Wesley, 1998.

Tore Dybå was born in 1961 and received his M.Sc. in Computer Science and Telematics from the Norwegian Institute of Technology (now NTNU) in 1986. He worked as a consultant for eight years both in Norway and in Saudi Arabia before he joined SINTEF in 1994. In addition to being a Research Scientist at SINTEF, he is also a Research Fellow at the Norwegian University of Science and Technology working on a Ph.D. thesis in Software Process Improvement (SPI). The focus of his Ph.D. thesis is an investigation of the key learning processes and factors for success in SPI.

Nils Brede Moe was born in 1972 and became a M.Sc. at the Norwegian University of Science and Technology (NTNU) in 1998. His main research areas in the field of Software Processes Improvement include: Measurement based improvement, assessments and improvement on an organisational level. Other research areas are Human-Computer Interaction (HCI) and E-commerce.

Philosophies and Approaches to Software Process Improvement

Yingxu Wang and Graham King*

IVF Centre for Software Engineering
Argongatan 30, S-431 53 Molndal, Sweden
Tel: +46 31 706 6014, Fax: +46 31 27 6130
E-mail: Yingxu.Wang@ivf.se or Yingxu.Wang@acm.org

*Research Centre for Systems Engineering
Southampton Institute, Southampton, SO14 0YN, UK
Tel: +46 31 706 6014
E-mail: Graham.King@solent.ac.uk

Abstract

Software process improvement (SPI) has been a widely accepted technology in software engineering research and in the software industry. This paper reviews existing methodologies for SPI, and explores alternative SPI methodologies. Philosophies behind the SPI methodologies are described. A set of generic rules and procedures of SPI are formally presented.

Key Words: Software engineering, software engineering process, software process
improvement, philosophies, approaches, methodologies, rules

1. Introduction

Concepts and methodologies of software process improvement (SPI) have been largely inspired by the work in management science, particularly in quality system principles and enterprise reengineering research. Shewhart [1] developed the concept of plan-do-check-act iteration. Later this concept was extensively applied in the Japanese manufacturing industry known as KAIZEN method [2], and was extended and interpreted by Deming known as the Deming cycle [3].

A number of SPI methodologies have been developed in the last two decades. These methodologies can be categorised into two types: model-based and benchmark-based SPI. The model-based SPI has been developed by Humphrey [4], Paulk et al. [5,6], Curtis et al. [7], Basili [8], Kuvaja et al. [9], ISO/IEC TR 15504-7 [10], and Wang et al. [11,12]. The benchmark-based SPI has been developed by IBM [13], IBM/IVF [14], and Wang et al. [12,15].

In practice, an SPI project starts by mapping of a software organisation's existing processes to a process model that is chosen for an assessment. The usual cases are that a software development organisation has only some loose and informal practices in software development, rather than a defined and coherent process system. However, a rigorous process-based software engineering has to start from process establishment, rather than process assessment in a software development organisation. Therefore the right order towards software process excellence in an organisation is first process establishment, second process assessment, and then process improvement, as shown in Figure 1.

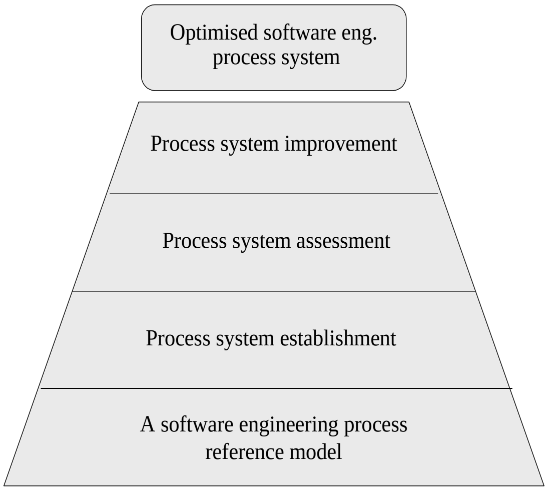

Optimised software eng. process system
Process system improvement
Process system assessment
Process system establishment
A software engineering process reference model

Figure 1. Process-based software engineering

In the following sections, the common rules of SPI are presented, philosophies for SPI is contrasted, and SPI methodologies that follow different philosophies are explored.

2. Rules of Software Process Improvement

Software process improvement is a complicated, systematic, and highly professional activity in software engineering that requires theory and models, skilled technical and managerial staff, as well as motivated top management commitment. A set of basic rules that predominate over software process improvement is summarised as follows:

Rule 1: Software process improvement is complicated system engineering.

A process improvement programme has to be thoroughly planned. There will be little achievement if an organization attempted to improve only a part of the many identified necessary processes in order to improve the whole process system and its performance.

Rule 2: Software process improvement itself is a goal-driven and a continuous process.

It is goal-driven because process improvement should have pre-determined goals and pre-designed approaches to achieve these goals. It is a continuous process because the trace of software process improvement is spiral-like and endless. During a software process improvement programme, the goals may aim to higher, organizational requirements may dynamic, and implement complicity may be increasing. Therefore, there is no absolute final end for a process improvement.

Rule 3: Software process improvement is an experiment process.

Empirical improvement recommendations and rules of thumbs should be treated as hypotheses. Impact and effectiveness of process improvement activities should be monitored and checked by periodical process reviews and/or assessments.

Rule 4: Software process improvement is risk-prone.

With regard directly to Rule 3, it can be seen that risks are naturally attached to any process improvement activities. Therefore, process improvement risks and potential impacts on other processes for an improvement should always be predicted. Also, risks for not implementing required improvement for identified problems should be estimated.

Rule 5: Software process improvement is a time varying system.

Process improvement is working in a dynamic environment, for varying application domains, and fast changing technical platforms. This means there is no specific model one can completely copy; and no specific methodology one can always follow. Therefore, model and methodology adaptation is always required in process improvement.

Rule 6: Software process improvement is a random system dominated by
human factors.

Parallel to Rule 5, process improvement is carried out by human being. The features of human factors in software engineering are mainly of diversity and goal-orientation.

A basic assumption is that a skilled software engineer as individual is an intelligent unit in a software engineering process, who would automatically adjusting one's activities to an optimising goal in the system.

It is noteworthy that process improvement solutions for an identified problem would be multiple; implementation for an recommended solution would be achieved by multiple approaches; and times and effort spend on implementation of an approach would be varying greatly by different individuals or teams. All these varying human factors should be taking into account in a plan of process improvement.

Rule 7: Software process improvement has preconditions.

Process improvement requires formally defined, established and experienced process systems. Process improvement on virtual processes has been proven wasteful.

Software process improvement can only be started based on established software processes. Otherwise, it's effect would be virtual if the process system is virtual itself.

Rule 8: Process improvement is based on process system reengineering.

Process system reengineering is the kernel of SPI. Reengineering can be carried out by: a) enhancing a process; b) changing a process; c) adapting a process; d) merging processes; e) cancel a process; and f) re-organising a process system.

Rule 9: Software process improvement achievement is cumulative.

At all above technical, organizational and cultural costs, the benefits of process improvement achievement, fortunately, can be cumulated permanently, if an organization continuously pursues software process improvement in a systematic and consistent way.

3. Philosophies in Software Process Improvement

In this section we describe the philosophies that behind the process improvement methodologies. There are various philosophies towards SPI. Key categories of SPI philosophy are as follows:

- Goal-oriented process improvement
- Operational process improvement
- Continuous process improvement

3.1 Goal-Oriented SPI

Definition 1. Goal-oriented process improvement is an approach by which a process system's capability is improved towards a predefined goal, usually a specific process capability level.

This approach is based on a simple and the most widely adopted philosophy to SPI: "The higher the better." For example, ISO 9001 provides a pass/fail goal with a basic set of requirements for a software process system. CMM, ISO/IEC 15504, and SEPRM (the Software Engineering Process Reference Model [12]) provide a 5/6-level capability scale, which enable software development organisations to set more precise and quantitative improvement goals.

Related approaches to goal-oriented SPI based on measurements and metrics can be refereed to the Goal/Question/Metric (GQM) paradigm [13, 14] and AMI method [15].

3.2 Operational SPI

Definition 2. Operational process improvement is an approach by which a process system's capability is improved towards an optimum profile, rather than the maximum capability level.

This is a realism philosophy towards SPI, that can be interpreted as "the smatter the better." It was argued that for maintaining sufficient competence, a software organisation do not need to push all its software engineering processes to the highest level, because it is not necessary and not economic. This philosophy provides a different thought on the idea that the higher the better for process capability as presented in the goal-oriented process improvement approach.

By the operational improvement approach, an optimised process improvement strategy is to identify the most sufficient (the minimum required) and economic target process profile, which provides an organisation the sufficient margins of competence in each process, but not necessarily sets them all at the highest level of a capability scale.

To illustrate this philosophy, we take the following historic story as an example. There was a well known story about King Qi's horse racing in China about 2000 years ago. King Qi had the best horses in his kingdom. He liked horse racing very much and he expected to win every time. However, once he lost to Ji Tain, a wizard of that time.

The horses were categorised in three classes, i.e., for the King: K_1, K_2, and K_3; and for Tian: T_1, T_2, and T_3. In the first match, they raced in the way: $K_1 – T_1$, $K_2 – T_2$, and $K_3 – T_3$. Not surprising, the King had won, as shown in Figure 2, because he had the best horses in each class.

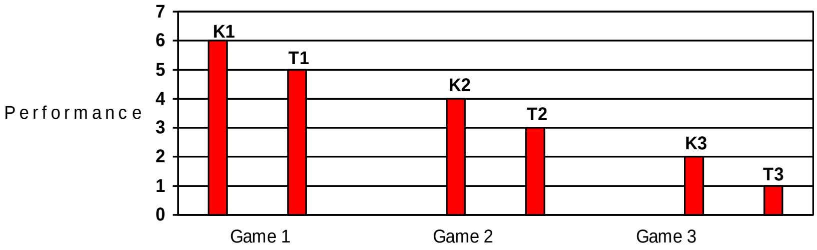


Figure 2. Example for decision making in process improvement (1)

However, in the second match, the wizard changed his strategy. Tian used his third class horse (T_3) against King Qi's first class (K_1), and of cause allowed the King win again for the first game. Then in the following two races, Tian used his first (T_1) and second (K_2) class horses against the King's second (K_2) and third (K_3) class horses respectively. Eventually Tian won the three-game racing for the first time in the history of the kingdom, as shown in Figure 3.

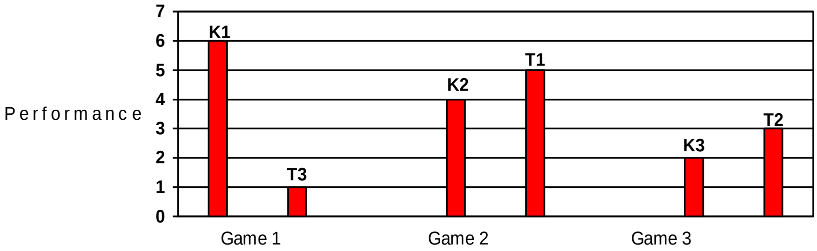


Figure 3. Example for decision making in process improvement (2)

This story provides a useful operational strategy in decision making for process improvement. That is, for software process improvement, an organisation is not necessarily to get all of its processes at the highest level to be competitive, because it would not be the best, most feasible and most economical solution for the organization. Instead, the best solution is just to have a marginal competitive in each process than the competitors. This inspires a new approach to software engineering process improvement – the benchmark-based process improvement, which will be developed in Section 4.

3.3 Continuous SPI

Definition 3. Continuous process improvement is an approach by which a process system's capability is required to be improved continuously, and towards better performance in all the time.

This is a philosophy towards ideal optimisation and perfection for a set of systematic activities. Continuous process improvement has been proven effective in engineering process optimisation and quality assurance. By this approach, SPI is a continuous spiral-like procedure.

In his work on 'Statistical Method from the Viewpoint of Quality Control,' Shewhart [1] established the statistical foundations of generic quality control system. He developed the concept of 'plan-do-check-act' iteration. The statistical quality control approach has largely influenced today's software process capability modelling and software metrics studies, as a proof that almost all of major current process models require systematic data collection and recommend quantitative process improvement.

Deming's work [3] drawn the attention of researchers and industrial practitioners for both quality and productivity. He proposed the approach to TQM. TQM is a management philosophy for achieving quality improvement by creating a quality culture and attitude throughout the entire organisation's commitment and involvement. This approach has been widely accepted in the manufacturing industry, and has been presented in the ISO 9000 standards.

Both statistical quality control and TQM have been extensively applied in the Japanese manufacturing industry. Based on these a 'KAIZEN' method was developed in the 1980s in Japan. Imai [2] supposed the term 'KAIZEN' as the key to Japan's competitive success in its manufacturing industry. 'KAIZEN' is, actually, two Chinese characters (Gai-Shan). 'ZEN' means good, satisfactory, or perfect; 'KAI' means change, update, or reform. Therefore 'KAI-ZEN' simply means to make better. Whilst its internal philosophy implies gradual and continuous improvement and/or attaining perfection. This is perhaps the most influential philosophy that has been widely accepted as one of the important quality principles as those of statistical quality control and TQM.

In continuous process improvement, there is no end for process optimisation, and all processes are supposed to be improved along the time. There is argument that the goals for improvement are not explicitly stated in this philosophy. Therefore, when adopting continuous process improvement, the top management should make it clear what are the current goal, as well as the short, middle and long term goals in a continuous process improvement pursuit in a software development organisation.

In empirical software engineering, goal-oriented SPI methodologies will still be in the main stream. While with the 2-D process models like ISO/IEC 15504 provide more precise process assessment results, and the benchmark-based process models provide empirical indications of process attributes, the operational process improvement, especially the benchmark-based SPI, will gain wider application. Also, the continuous process improvement approach will provide sustainable long-term strategic planning for software process improvement.

4. Methodologies for Software Process Improvement

Based on the above discussion of the philosophies for process improvement, this section investigates possible SPI methodologies in software engineering. Two basic SPI methodologies – the model-based and benchmark-based process improvement will be explored. The former improves a process system from a given level in a defined scale to a higher level; The latter provides improvement strategies by identifying gaps between a software development organisation's process system and a set of established benchmarks. In addition, a combined approach of the above can be adopted.

4.1 Generic Approaches to SPI

A generic procedure of SPI has been identified in [12] with 6 steps as follows:

1) Examine the needs for process improvement;
2) Conduct a baseline assessment;
3) Identify process improvement opportunities;
4) Implement recommended improvement;
5) Review process improvement achievement;
6) Sustain improvement gains.

In the following sections, we will explain how the generic process improvement procedures are implemented by different techniques. Three software process improvement methodologies, such as model-based, benchmark-based, and integrated SPI, will be presented as follows.

4.2 Model-Based Improvement

Definition 4. Model-based improvement is an SPI methodology by which a process system of a software development organisation is improved based on its performance and capability profile provided by a model-based assessment.

Model-based improvement is a kind of absolute improvement approach. By this approach, processes of a software development organisation are suggested for improvement according to a process system model step-by-step. CMM, BOOTSTRAP, and SEPRM are examples of such model-based process improvement methodology.

There is also a special case in model-based improvement - the standard-based SPI. By this approach, processes of a software development organisation are suggested for improvement according to a standardised process system model. ISO/IEC 15504 provides a standard-based improvement method. However, it is noteworthy, that ISO 9001 is probably not suitable for being a standard-based improvement method, because it lacks a process improvement model and a step-by-step improvement mechanism.

Referring to the generic procedure of SPI, when the baseline profile of a software organisation is obtained, and process strengths and weaknesses are analysed, the potential process areas for improvement and priorities can be identified following the method provided in Table 1.

Table 1. Improvement opportunities analysis

Input	Method	Output
• Processes to be assessed and target capability levels; • Process capability profile as derived in the assessment; • Process strengths and weaknesses analysis; • The SEPRM process reference model.	• Analysis of process improvement opportunities will be conducted according to the generic procedures; • Identify improvement priorities of each process by evaluating the gap to the target capability level. The improvement priority will be described as high [H], medium [M], low [L], or No [N]; • Analyse and describe impact and potential risks that may be risen in an improvement activity.	Process improvement opportunities analysis.

A case study on SEPRM-based process improvement is shown in Table 2. In Table 2 the criteria adopted for classifying the improvement priority, *IP*, can be formally derived by the following formula:

$$
\begin{aligned}
IP \quad &= H, \quad \textit{Weakness} > \textit{1 capability level;} \\
&= M, \quad \textit{Weakness within 1 capability level;} \\
&= L, \quad \textit{Strength} < \textit{0.3 capability level, which would} \qquad (1) \\
&\qquad\quad \textit{be sensitive when capability turbulent;} \\
&= N, \quad \textit{The rest, which have no improve requirement with}
\end{aligned}
$$

regard to the specified target capability level.

Different thresholds would be defined in Formula 1 for a specific process improvement case.

Table 2. Sample Template for Process Improvement Opportunities Analysis

No.	**Process**	**Strengths(+)/ Weaknesses (-)**	**Improvement Priority (IP)**	**Remarks and Risks**
1	Organisation			
1.1	Organisation structure category			
	......			
1.2	Organisational process category			
	......			
1.3	Customer service category			
	......			
2	Development			
2.1	Software engineering methodology category			
2.1.1	Software engineering modelling	0.1	L	
2.1.2	Reuse methodologies	-0.4	M	
2.1.3	Technology innovation	-0.9	M	
2.2	Software development category			
2.2.1	Development process definition	1.5	N	
2.2.2	Requirement analysis	0.4	N	
2.2.3	Design	0.3	N	
2.2.4	Coding	0.4	N	
2.2.5	Module testing	0.3	N	
2.2.6	Integration and system testing	0.1	L	
2.2.7	Maintenance	-0.1	M	
2.3	Software development environment category			

2.3.1	Environment	0	L	
2.3.2	Facilities	1.2	N	
2.3.3	Development support tools	-0.5	M	
2.3.4	Management support tools	-0.8	M	
3	Management			
3.1	Software quality assurance category			
	......			
3.2	Project planning category			
	......			
3.3	Project management category			
	......			
3.4	Contract and requirement management Category			
	......			
3.5	Document management category			
	......			
3.6	Human resource management category			
	......			

4.3 Benchmark-Based SPI

Definition 5. Benchmark-based improvement is an SPI method by which a process system of a software development organisation can be improved based on its performance and capability profile provided by a benchmark-based assessment.

Benchmark-based improvement is a kind of relative improvement approach. By this approach, processes of a software development organisation are suggested for improvement according to a set of process benchmarks. As described in Section 3.3, benchmark-based process improvement is supported by the operational process improvement philosophy. It would provide an optimised and economical process improvement solution in practice.

A benchmark-based process improvement will be carried out in 6 steps as described in the generic improvement procedure. With regards to the model-based process improvement methodology, the features of a benchmark-based process improvement are as follows:

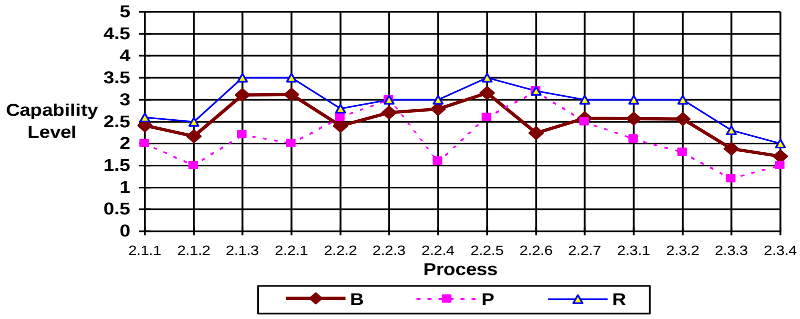
- The philosophy for a benchmark-based process improvement is to ‘filling the gaps’ rather than ‘the higher the better’ as that in a model-based process improvement;

- The improvement opportunities are identified based on gap analysis between the plotted process profile and the benchmarks;

- The improvement priorities are determined by quantifying the degree of gaps between the plotted process profile and the benchmarks;

- The improvement achievement is evaluated by checking if the gaps have been reduced, and if the process capabilities have been enhanced marginally above the process benchmarks.

A case for demonstrating an organisation’s baseline and improved capability profiles in a benchmark-based process improvement is shown in Figure 4. Figure 4 shows, the baseline process capability profile (P) of an organisation has been improved to an adaptive process profile (R) that is marginally above and along with the benchmarked curves (B).



B – Benchmark curve; P – Baseline process profile; R – Improved process profile

Figure 4. SEPRM benchmarks-based process improvement

Note in Figure 4 only the development process benchmarks and profiles were shown. Adopting the SEPRM benchmarks provided in [12], the organisation and management process subsystems improvement can be carried out in the same way.

4.4 Integrated SPI

Definition 6. Integrated improvement is a combined model-and-benchmark-based SPI method by which a process system of a software development organisation is improved based on its performance and capability profile provided by an integrated model-and-benchmark-based assessment.

The integrated process improvement method inherits the advantages of both absolute and relative SPI methods. By the integrated improvement method, processes of a software development organisation are suggested for improvement according to a benchmarked process system model. SEPRM is the first benchmarked model for enabling integrated model-and-benchmark-based process improvement.

5. Conclusions

This paper has reviewed the historical development of software process improvement methodologies and philosophies, and its connection with quality system principle research in management science. A set of basic rules of SPI has been developed. The generic procedure and preconditions of SPI have been explored. Different philosophies and corresponding methodologies are contrasted in the context of process improvement.

References

[1] Shewhart, W.A. (1939), *Statistical Method from the Viewpoint of Quality Control*, The Graduate School, George Washington University, Washington, D.C.

[2] Imai, M (1986), *KAIZEN: The Key to Japan's Competitive Success*, Random House, New York.

[3] Deming, W. E. (1986), *Out of the Crisis*, Massachusetts Institute of Technology Press, Cambridge, Massachusetts.

[4] Humphrey, W.S. (1988), Characterising the Software Process: A Maturity Framework, *IEEE Software*, March, pp.73-79.

[5] Paulk, M.C., Weber, C.V. and Curtis, B. (1995*), The Capability Maturity Model: Guidelines for Improving the Software Process*, SEI Series in Software Engineering, Addison-Wesley.

[6] Paulk, M.C., Curtis, B. and Chrissis, B. (1993), Capability Maturity Model, Version 1.1, *IEEE Software*, July, pp.18-27.

[7] Curtis, B., Kellner, M.I., and Over, J. (1992), Process Modelling, *Communications of the ACM*, Vol.35, No.9, pp.75-90.

[8] Basili, V. (1993), The Experience Factory and Its Relationship to Other Improvement Paradigms, *Proceeding s of 4th European Software Engineering Conference*, LNCS 717, Springer-Verlag, pp. 68-83.

[9] Kuvaja, P., Simila, J., Kizanik, L., Bicego, A., Koch, G. and Saukkonen, S. (1994), *Software Process Assessment and Improvement: The BOOTSTRAP Approach*, Blackwell Business Publishers.

[10] ISO 15504 ISO/IEC TR 15504-7 (1998), *Information Technology – Software Process Assessment - Part 7: Guide for Use in Process Improvement*, ISO/IEC, Geneve, pp. 1 - 36.

[11] Wang, Y., Court, I., Ross, M., Staples, G. and King, G. (1996a), Towards a Software Process Reference Model, *Proceedings of International Conference on Software Process* Improvement (SPI'96), Brighton UK, pp.145-166.

[12] Wang, Y. and King, G. (1999), *Software Engineering Processes: Principles and Applications*, CRC Press, USA.

[13] Basili, V.R., Caldiera, C., Rombach, H.D. (1994), Goal Question Metric Paradigm, *in Encyclopedia of Software Engineering (Marciniak, J.J. ed.)*, Vol.1, John Wiley & Sons.

[14] Solingen, R. V. and Berghout, E. (1999), *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*, The McGraw-Hill Co., London.

[15] Pulford, K., Combelles, A.K. and Shirlaw S. (1996), A Quantitative Approach to Software Management: The AMI Handbook, Addison-Wesley Publishing Co.

[16] IBM (1996), *Ensuring Profitable Investment in Software Process Improvement*, IBM Technical Report.

[17] Wang Y., Wickberg, H. and Dorling, A. (1999e), Establishment of a National Benchmark of Software Engineering Practices, *Proceedings of 4th IEEE International Software Engineering Standards Symposium (ISESS'99)*, IEEE Computer Society Press, Brazil, May, pp.16-25.

[18] Wang Y., King, G., Dorling, A., Ross, M., Staples, G., and Court, I. (1999), A Worldwide Survey on Best Practices Towards Software Engineering

Process Excellence,
ASQ Journal of Software Quality Professional, Vol.2, No3, Sept. 1999

The General Effect of an Integrated Software Product Evaluation

Jozsef Gyorkos, Ivan Rozman, Robert T. Leskovar
[1] University of Maribor, Faculty of Electrical Engineering and Computer Science,
Smetanova 17, SI-2000 Maribor, Slovenia; e-mail: gyorkos@uni-mb.si

Abstract

Well organized software product evaluation has a general effect to the discipline of the software processes. In our paper a case of the governmental institution where step-by-step introduction of formal evaluation methodologies has been forced to the outsourced projects is presented. The strategic influence of outsourcing is stressed throughout and the methods of risk prevention with improved communication of the outsourced client and vendor described. Risk prevention is based upon the formal controllability of outsourced projects. Formality is achieved by using, firstly, project management environments, which allow communication between parties, whereby it is not only a planning/tracing tool. Secondly, there are process and product assessment mechanisms.

I. Introduction

Outsourcing is an important method of managing information systems (IS). In relation to government projects, it is now a common practice, primarily when there is a requirement for the development of different applications, integrated into a comprehensive system.

Changes in the labour market have brought even more intensive outsourcing. Recently companies have moved away from traditional long term employment arrangements (insourcing) to relatively short-term market mediated arrangements (outsourcing) [Slaughter, 1996]. This source states (Derived from research in U.S.), that firms in the public sector are more likely to outsource IS employment, than firms in the private sector.

Our paper will use as a basis for the discussion the Government Information Centre of the Republic Slovenia, the CVI.

Outsourcing has brought about dynamic boundary changes, i.e. (distributed application development, non transparent organisation structure of vendors and heterogeneous methodologies). However, the continuos operation of an integrated information system, must be assured at a conceptual level, being in relation to the data and common functional structures. The structure is based on the rules and processes of public administration, but at the same time it is highly dependent on actual decisions, rules and policies of the State and (actual) Government. Outsourcing also circulates the resources (money) received from the tax payer, back to its productive source - the tax payer.

CVI has a very broad aim, that is the introduction and maintenance of information technology in all government institutions, and to lesser degree public institutions.

Slovenia is small country in relation to many other European states, and like others born or 'reborn' in the late 1980s and early 1990s, it has had to develop a whole new IS supported administrative apparatus, which is a requirement for a modern functional state. In the world of computer technology, there is a similar situation, whereby, there are not so many clients, but the functionality of the server is the same as in larger states.

II. Types and Extent of Outsourcing

Outsourcing is the contracting of various systems to outside information systems vendors [Nam, 1996]. For most managers – IS outsourcing clients - outsourcing sounds like 'entropy' – the possibility for disorder an uncertainty in a system. Consequently, outsourced projects can be endangered by 'entropy', if they are not managed in a proper fashion. The minimal attributes of 'properly' managed IS are those well known from software process models - best illustrated in the key process areas of Capability Maturity Model [Paulk, 1995]. For example, at least the second level key process areas must be covered: requirements management, project planning, configuration management, project tracing and oversight, quality assurance, subcontract management.

Nam mentions four types of outsourcing when he talks of the relationship between 'Extent of substitution by vendor' and 'Strategic impact of applications', these are: support, reliance, alignment and alliance. Using the CVI as a model we can see the distribution of outsourcing contracts among different types of outsourcing in Table I below.

In non-information technology industries, 'alignment' and 'alliance' are the usual types of outsourcing. In the forthcoming information society, the business process in a given public administration will be under pressure to re-engineer, due to the process automation and the demand for information dissemination, with or without administrative assistance.

The low level of alignment and almost non-alliance in outsourcing explains the strategic position of CVI. Whereby, we have not only a reliable information technology (IT) service provider for public administration but also the added responsibility of having a forum for making long-term, highly influential decisions in IT.

TABLE I
TYPES OF OUTSOURCING AND
THEIR APPROXIMATE EXTENT IN CASE OF CVI

Type of Outsourcing	*Tasks (e.g.)*	*Distribution of Outsourcing Contracts**
1. Support (non-core IS activities, small contracts)	contract programming, hardware maintenance, minor technical services, installations etc.	50%
2. Reliance (large extent of substitution by vendors in non-core IS activities)	same activities and tasks as above, longer length of contracts	30%
3. Alignment (low extent of substitution and high strategic impact)	consulting, technical supervision for IS planning and design, small contracts with strategic influence	20%
4. Alliance (high extent of substitution by vendor and high strategic impact)	substitution of in-house IS operations and vendors' responsibility for highly strategic IS activities, based on mutual relationships, highest commitments from vendors and clients	almost 0%

* *Derived out of values of contracts.*

III. SUPERVISION THRESHOLD

Preparation and *control* of the Outsourced projects performed by client, are both highlighted in this paper.

The scope of *preparation* is the requirements definition and selection of the vendor (usually directed by the legislative council). The outsourcing vendor is the organisation which delivers the required product and/or service. In this paper we will focus on outsourced projects where software develops within information technology projects and all IT related services.

Control of the Outsourced project by client consist of project tracing, quality assurance and of change management (technical or contextual – business process - changes).

When an outsourced project is running, threshold must be drawn on the scale, representing the degree of supervision of the vendor by the client. Table II highlights the achievements and drawbacks for a client regarding the different degrees of supervision.

IV. Sources of Risk

Different classifications of risk can be found in the literature [Boehm, 1989; PRINCE, 1995] but the aim of all methodologies is the prevention of project being endangered. Risk management is an built-in part of practically all software development methodologies and quality assurance techniques (at least indirectly). The different phases of the software process, or software project development, have the effect, whereby, risk management appears in many forms and with different scopes of influence. Table III highlights this.

TABLE II
DEGREE OF CLIENT SUPERVISION IN OUTSOURCED PROJECTS

Type of supervision	*Supervised products*	*Achievements for Client*	*Drawbacks for Client*
minor	general specification, contract, integration test results of the final product	minimal in-house resources needed	unexpected results
managed	as in minor, project tracing results (schedule, resources)	dependable resource management	technical imperfection can turn up
technical	as in minor, technical specification, change management records, test coverage plans and test results	technical overview, reliable products	highly-specialised technical staff required
optimal	minor, managed and the technical joined	early uncertainty detection, reliability	resource pretentious, asking about worth of outsourcing

V. Evolving Practice

Three years ago within our environment of organised activities of risk prevention in outsourced projects, there was the introduction of an overall quality system based on ISO 9001, with the addition of a Capability Maturity Model. We gave a special attention to those processes that contributed to the overall control of (1) the outsourcing vendors and their processes, (2) the products delivered by vendors and (3) the activities of the client.

It is important to stress, that these processes not only control the vendor but also contribute to a flexible and transparent project management in relation to the client (in our case the CVI). The intensity of project management activities, carried out by a client, depends on the type of supervision, formulated in Table II. Table IV, enumerates the existing practices within CVI in relation to risk prevention.

A. Assessment of the process

The most important aspect of the assessment of the process is, when a new outsourcer is chosen i.e. - no previous experience with him or her. This assessment can now be carried out using a different criteria, usually CMM or ISO and this is done before a formal agreement between vendor and client is agreed. PROCESSUS methodology is highlighted in [Györkös, 1996; Rozman, 1997]. This approach is also mentioned in Table III, which joins both approaches in an comprehensive tool supported questionnaire. Figure 1 shows the logical object model of the tool. However assessment of any process as a means risk prevention, must be done also when the project is in progress, with checking of existing practice and of an active quality assurance model established.

TABLE III
MEANS OF RISK MANAGEMENT IN THE SOFTWARE PROCESS.

Phase/Part of the Process	*Critical Success Factors*	*Means of Risk Management*
Requirement definition	formality of requirements specification (RS) compliance of RS with - long term strategy - actual needs	Audit and Review of specification
Procurement/ acquisition	credibility of outsourcer in public competition	Audit, reference and previous experience (PROCESSUS Tool, chapter 5.1)
phase of development	methodological approach enabling traceability and technical quality	Technical review Project tracing
Release	product quality (using the parameters derived from ISO 9126)	Product assessment (PRO+ Tool, chapter 5.2)

When consistent project management is used only the ‘products’ of the project management process can be checked. In our case the consistency of project management, ensures the adopted PRINCE methodology [PRINCE, 1996]. This PRINCE methodology is applied in the form of ProjectOffice, a distributed environment with functionalities, i.e. project group co-ordination, process control, exception handling and quality assurance.

B. Assessment of the product

Review is an activity that checks the correctness of particular software development phase at technical level. In the context of an ‘outsourcing review’, this is rarely done by the outsorcing client, but by the vendor himself. The vendor is obliged to perform the quality assurance of his products at a technical level. The client checks periodically whether the partial results are correct, this results are then published within progress reports by the ProjectOffice.

Before a release, the customer (CVI) performs one final assessment of the product not only from general view. This assessment can take up to five days, depending on the size of the product and the level of its integration with other (existing or in-development) products. These results are then collated in a tool called PRO+ which is planned to be a consistent part of the ProjectOffice application.

PRO+ is based on quality attributes described in standards; ISO 9126 (IT – Software product evaluation – Quality characteristics and guidelines for their use), and partialy on ISO 9127 (Information processing systems – User documentation and cover information for consumer software packages) and ISO 12119 (IT – Software packages – Quality requirements and testing). In the background of the tool a formal decision support is defined, this enables calibration of parameters and in the process will give reports of the different views (client, developer, user). The calibration is fully transparent and enables multiple-level assessment of the same product.

TABLE IV
EXISTING RISK PREVENTION PRACTICES IN CVI

Activity/task, subject of the outsourcing contract	Type of outsourcing (according to Table I)	Method	Tool	Practice
application development (full life cycle, technical level)	reliance	weak, TBD	yes	good
programming/prototyping	support reliance	strong	yes	good
procurement/acquisition	-	strong, BOR	yes	good
strategic plans preparation	alignment (alliance)	weak	none	medium
IS planning and design	alignment	medium, TBD	weak (different tools)	
Maintenance	reliance	none	medium (interactive database apllication)	improving
project management	alignment	strong	yes (ProjectOffice)	good
quality assurance (technical supervision including acceptance testing and validation)	support, alignment	strong	PROCESSUS PRO+	improving from medium

TBD – to be declared (method preparation/selection in progress).
BOR – based on legislative regulation.

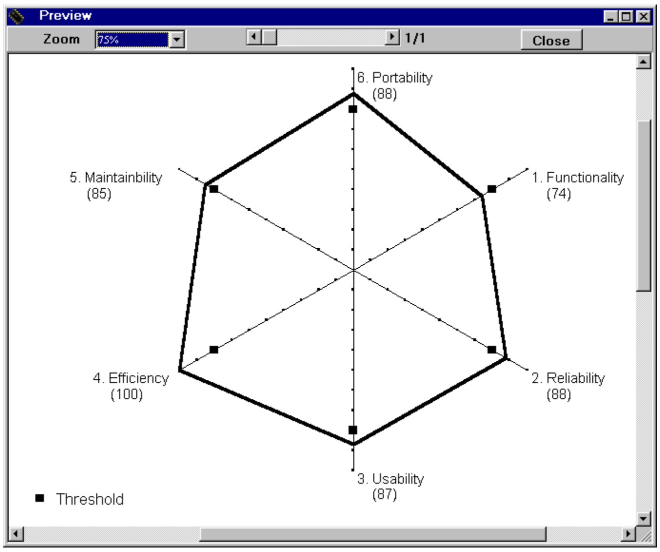


Fig. 1. High-level report from PRO+ Tool.

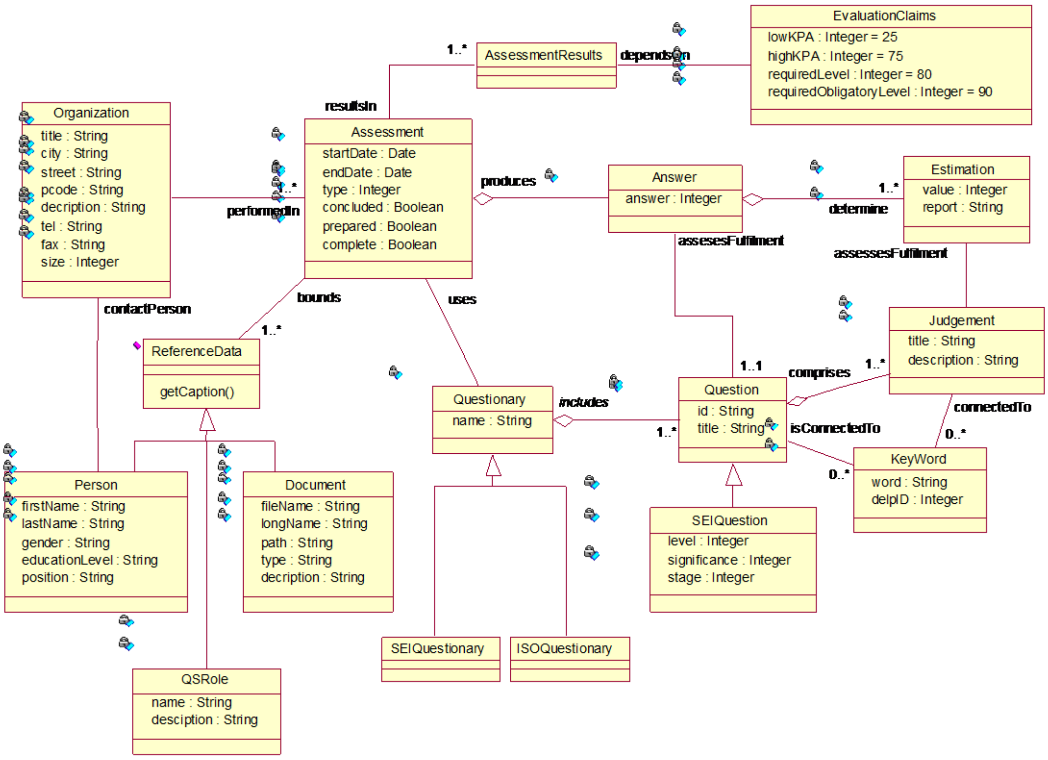


Fig. 2. The logical object model of the PROCESSUS Tool.

The results of the final decision (assessment of the product), are given in the form of a vector, containing multiple parameters, with normalised values (from 0 to 1). Each parameters (functionality, reliability, usability, efficiency, maintainability, portability and completeness of the documentation) is kept on its own ordinary axis, in the form of a Kiviat diagram (Figure 1).

VI. CONCLUSIONS

This paper has described the various practices and problems involved in a large governmental institution, and how to solve them. Client-vendor relationships in outsourced projects, are often limited to a project initiating and a product delivery.

Initially we listed the various types of outsourcing, and declared the degree of client supervision in various projects. A practice based on mutual project management and methodology-approach to the quality assurance. With three years of experience, practicing this approach, we can show an essential progress in the controllability of outsourced projects. This has resulted in a more reliable risk prevention, better product quality, resource optimisation and not least of all, in better communication and co-operation with vendors.

REFERENCES

[Boehm, 1989] B.W. Boehm, Software Risk Management, IEEE Computer Society, 1989.

[Györkös, 1996] J. Györkös, I. Rozman, R. Vajde-Horvat, M. Hericko, "Quality Management in Software Development Process: An Empirical Model", Proceedings of IEEE International Engineering Management Conference, IEEE Press, 1996.

[Paulk, 1995] M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber, Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-24, February 1995.

[Slaughter, 1996] S. Slaughter, S. Ang, Employment Outsourcing in Information Systems, Communications of ACM, Vol. 39, No. 7, 1996.

[Rozman, 1997] I. Rozman, R. Vajde-Horvat, J. Györkös, M. Hericko, "PROCESSUS - Integration of SEI CMM and ISO quality models", Software Quality Journal, Volume 6 Number 1 March 1997, ISSN 0963-9314, Chapman & Hall, 1997.

[Nam, 1996] K. Nam, S. Rajagopalan, R. Rao, A Chaudhury, "A Two-Level Investigation of Information Systems Outsourcing", Communications of the ACM, Vol. 39, No. 7, 1996.

[Paquin, 1996] J.P. Paquin, J. Coulliard, P. Paquin, D. Godcharles, "Earned Quality: Improving Project Control", Proceedings of IEMC'96, IEEE Press, 1996.

[PRINCE, 1995] Management of Programme Risk, CCTA The UK Central Computer and Telecommunications Agency, London, 1995.

[PRINCE, 1996] PRINCE 2, Project Management for Business, The Stationery Office Publication Centre, London, 1996.

Session 8
SPI Surveys

Chairman

Miklos Biro

Sztaki, Budapest, Hungary

Software Process Improvement Network in the Satakunta Region - *SataSPIN* -

Timo Varkoi & Timo Mäkinen
Tampere University of Technology,
Information Technology, Pori, Finland

Abstract

The core of the *SataSPIN* project is to help small and medium sized software enterprises to develop their operations using international software process models. The project uses ISO/IEC 15504 TR (SPICE) as the software process assessment and improvement framework. The main goal is to set up a software process improvement (SPI) program in each of the participating companies and thereby to establish a network of companies promoting good software practices in the region. The project is based on the co-operation of the participating enterprises. The main activities in the project are process assessments, improvement planning, and consultation and training to support the improvement activities. The project has offered a wide variety of courses and seminars in the area of software engineering and management.

At this point the companies have set their own priorities for SPI and first SPICE assessments have been performed. In the assessments all of the seven companies were involved, 11 projects and 27 SPICE-processes were assessed. Each company was assessed separately and the assessment results were reported in feedback sessions and in detailed assessment reports.

During the project special attention has been given to e.g. requirements traceability and measurement.

The second phase of the project will ensure the continuity of the SPIN and the active co-operation within the Satakunta software industry and enlarge the number of the companies participating the *SataSPIN* project.

Introduction

A project to establish a software process improvement network (SPIN) in the Satakunta region in Western Finland was started in August 1998. The project is called *SataSPIN*. The core of the project is to help small and medium sized enterprises (SMEs) in software business to develop their operations using international software process models. In the first phase of the project seven small software organisations are participating to improve their software processes. The project uses ISO/IEC 15504 TR (SPICE) as the software process assessment and improvement framework [2].

The project provides the participating companies with training and consultation on subjects related to software processes. An essential part of the consultation activity is the process assessments. The companies can get assistance also in planning and implementation of the improvements. Training activities within the project are targeted to support the improvement of the software processes and to enhance the competencies of the personnel. All the activities are tailored separately for each company to ensure flexibility in the participation and alignment with the business goals.

The organisations in the project have from 2 to 50 employees in software engineering related positions. Typically the personnel in the companies is professionally experienced and highly motivated in their work. All participating companies have distinguishable software products, are expanding their businesses and are eager to dynamically develop their operations. The most outstanding feature in these companies is their almost unconditional customer orientation. In the beginning of the project the organisations naturally had poor knowledge of software process models but, on the other hand, rich experience and good understanding of software process implementation. For the time being all companies have been assessed and are now working to plan and implement the improvements.

During the project special attention has been given to requirements traceability and measurement. The requirements management is an important process in ensuring a sound starting point for a software project. The measurement is often considered to be a key element in successful software process improvement (SPI).

The project is now moving to its second phase, in which more software companies in the Satakunta region will be involved. The experiences of the first phase will be used to assist the SPI efforts of the whole software industry in the region. In addition the project management experiences of the *SataSPIN* project will be disseminated.

Project goals

The main goal is to set up a software process improvement program in each of the participating companies and thereby to establish a network of companies promoting good software practices in the region. The second phase of the project will concentrate in increasing the number of the organisations participating, and distributing the experiences of the first phase. The potential in the area is estimated to be 40 enterprises and 400 professionals.

From the SME's point of view the benefits of the project include the improvement of customer satisfaction and competitiveness; the management of the growth of the business; the improvement of competence and motivation of the personnel; the development of working methods and the improvement of knowledge and skills by the software engineering training.

Project implementation

The project is based on the co-operation of the participating enterprises e.g. by sharing software process improvement experiences and best practices among the participants. The main activities in the project are process assessments, improvement planning, and consultation and training to support the improvement activities. General structure of the project is displayed in fig. TKVARKOI.1.

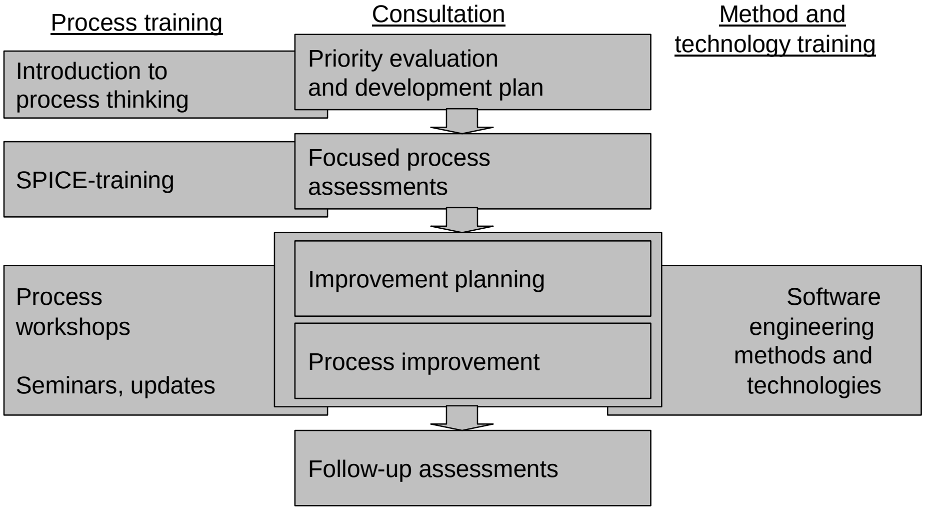

Process training
Consultation
Method and technology training
Introduction to process thinking
Priority evaluation and development plan
SPICE-training
Focused process assessments
Improvement planning
Process improvement
Process workshops
Seminars, updates
Software engineering methods and technologies
Follow-up assessments

Fig. TKVARKOI.1: *SataSPIN* - the general structure of the project

The project receives substantial public funding from the European Social Fund (ESF). The responsible organiser is Pori School of Technology and Economics (PoSTE) and the project's operative management is delegated to Software Process Improvement Center (SPIC). Project stakeholders are shown in fig. TKVARKOI.2.

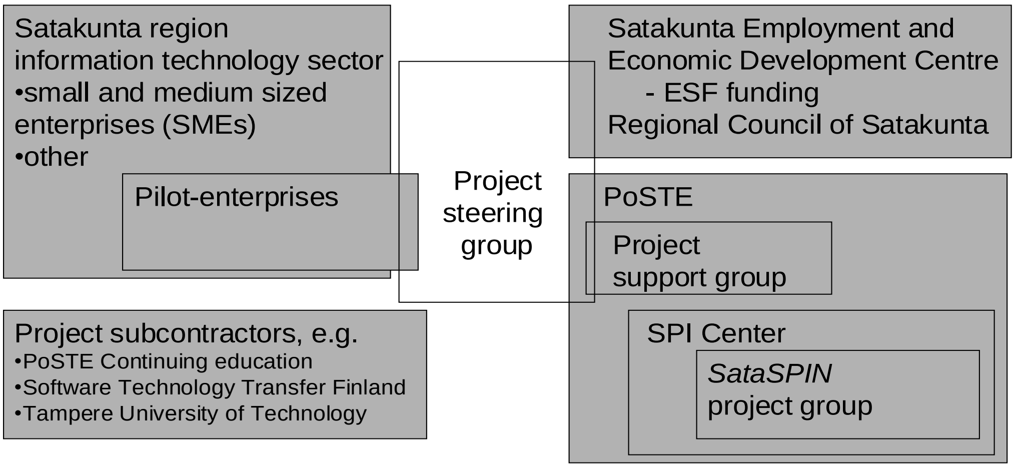

Satakunta region information technology sector
•small and medium sized enterprises (SMEs)
•other
Pilot-enterprises
Project steering group
Satakunta Employment and Economic Development Centre
- ESF funding
Regional Council of Satakunta
PoSTE
Project support group
SPI Center
SataSPIN project group
Project subcontractors, e.g.
•PoSTE Continuing education
•Software Technology Transfer Finland
•Tampere University of Technology

Fig. TKVARKOI.2: *SataSPIN* - the stakeholders

Results

At this point the companies have set their own priorities for SPI and first SPICE assessments have been performed. In the assessments all of the seven companies were involved, 11 projects and 27 SPICE-processes were assessed. The companies have participated actively in both training and consultation provided by the project. The training consists of software process improvement related training and training of software technologies and methods. By June 1999 the project had trained 145 persons with an average of 3,8 days each.

Consultation is mainly software process assessments, assistance in improvement planning and guidance in adaptation and implementation of single processes.

The companies are now working on process improvements and concrete, assessed SPI results are available after the follow-up of assessments.

Software process improvement background

Software process improvement (SPI) in small enterprises or business units requires special attention when applying models and standards which usually have been designed from the viewpoint of large organisations. Today the two publicly available, comprehensive models for software processes and process improvement are The Software Engineering Institute's Capability Maturity Model for Software (SW-CMM) [5] and the International Standards Organisation's ISO/IEC 15504 Technical Report - Software Process Assessment (SPICE) [3]. Both models can be applied to small organisations.

The SPI views of the small companies are not fully aligned with the SW-CMM maturity levels [7]. The SPICE Engineering (ENG) process category corresponds to the SW-CMM Software Product Engineering KPA on level 3, but the companies ranked it their first priority. Similarities can also be found, e.g. in Project Management, Requirements Management and Quality Assurance.

The differences can, to some extent, be related to the actual process capability of the company. For instance, a company with strong project management does not necessarily see it's strength and therefore may overlook the process. In small companies Subcontract Management could be a very small process as there is only a few suppliers, and small companies could also be very closely networked together. Few products and small number of employees can partly explain the relatively low priority of Configuration Management; only one person can be responsible for e.g. version control. On the other hand the individualism practised could direct the interest in testing within software engineering processes.

Similarities in the interviewed priorities and the models confirm the view that SW-CMM adapts to small organisations, too. On the other hand the strong customer-orientation of small companies also explains their interest in the processes like Requirements Management and Quality Assurance.

The type of business does not seem to affect the interests of small companies but clearly the size of the organisation does. A larger organisation of the small ones seems to be more interested in the management processes like Risk management and also in ensuring the product quality.

Software process improvement priorities

The project uses SPICE as the framework for software processes and also follows the eight-step cycle for continuous software process improvement described in SPICE Part 7 [4]. In step 1. Examine organisation's needs, we define the improvement goals for each company. This is done using BootCheck quick assessment tool [1]. The results support preliminary process improvement program planning and guide in selecting processes for the SPICE assessment. The assessment output is also used in detailed process improvement action planning.

As a part of the step 1, the representatives of management and engineering in the companies were asked to list the processes they would like to start the SPI with. The participants were specifically asked to think about processes that needed improvement first and would benefit the company most, and not the processes that are in general critical for the company. The results of the study are described in [7].

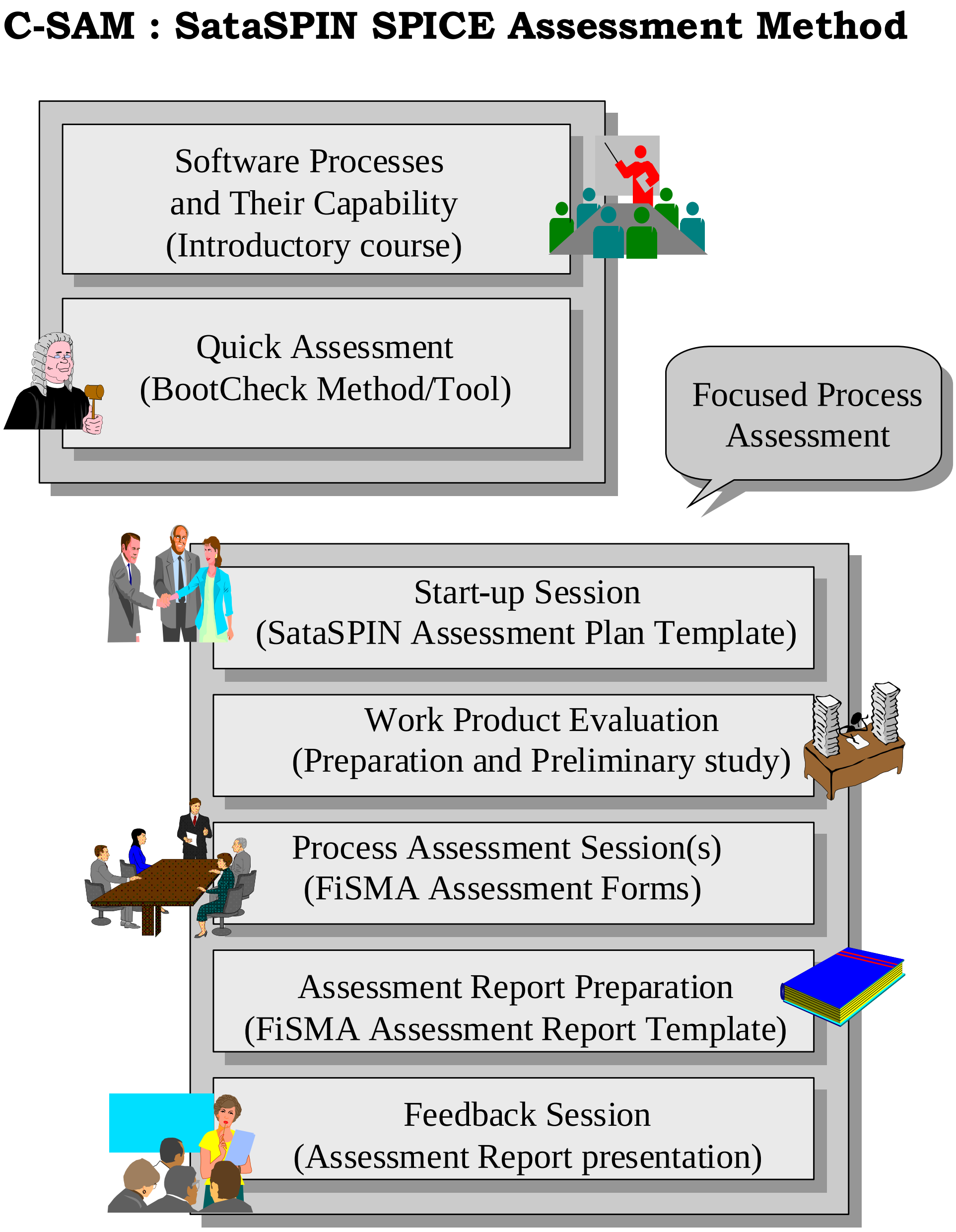
First the process categories were prioritised on a scale from 1 to 5 and then the most important processes within the categories were discussed and selected. The two most important process categories were considered to be Engineering (ENG) and Management (MAN), while Organisation (ORG) processes were of least interest.

The processes of common interest in the two most important process categories include ENG.1.2 Software requirements analysis and ENG.1.6 Software testing, and MAN.2 Project management and MAN.4 Risk management. Other notable processes of interest are CUS.3 Requirements Elicitation, SUP.1 Documentation, SUP.3 Quality assurance and ORG.2 Improvement process.

Process assessments

During the SPI project several software projects of the participating companies have been assessed. Each company was assessed separately and the assessment results were reported in feedback sessions and in detailed assessment reports. SPICE assessments are carried out using the assessment forms developed by Risto Nevalainen of STTF Oy and distributed by the Finnish Software Measurement Association (FiSMA). The assessment program is described in fig. TKVARKOI.3.

Assessment results and observations point out, for instance, that requirements management is based mostly on individual effort and that documented traceability practices are nearly non-existent. Even though it might be possible to manage very small projects without established requirements management, most often the outcome is an obscure scope of the customer project, slipping timetables and decreased profit. The assessments give lots of detailed information and improvement ideas to the assessed organisations [8].



C-SAM : SataSPIN SPICE Assessment Method

Software Processes and Their Capability (Introductory course)

Quick Assessment (BootCheck Method/Tool)

Focused Process Assessment

Start-up Session (SataSPIN Assessment Plan Template)

Work Product Evaluation (Preparation and Preliminary study)

Process Assessment Session(s) (FiSMA Assessment Forms)

Assessment Report Preparation (FiSMA Assessment Report Template)

Feedback Session (Assessment Report presentation)

Fig. TKVARKOI.3: *SataSPIN* - the assessment program

Requirements traceability

Requirements traceability is one of the subjects that the project has emphasised. Pressman [6] gives one definition of traceability: The ability to trace a design representation or actual program component back to requirements. This definition contains the idea of backward traceability: from work products to requirements to ensure that work products fulfil the requirements. On the other hand it is equally important to have forward traceability from requirements to work products to be able to ensure that all requirements have been satisfied.

Small organisations have typically difficulties to meet traceability demands presented in assessment models. Customer requirements are collected, but changes are not systematically managed. Requirements are not identified, which impedes traceability. Though positive exceptions exist, too, when requirements are managed throughout the project to satisfy both the customer and the supplier. In SPICE the traceability improvement is related to improving the process capabilities primarily of the engineering processes and their base practices on level 1, and to supporting it with the level 2 management practices. The essential improvement targets common to most small organisations are [8]:

- Identify the requirements and track changes
- Establish documentation and version control
- Reviews of customer requirements
- Checklists to assure quality of work products

Measurements

Another topic of interest during the project has been measurements [9]. Measurement is a key issue in software development process improvement as well as in developing software product quality. We collect and analyse data to be able to make better decisions. Measurement can be applied to all phases of software development lifecycle from collection of customer requirements to maintenance of the software product. Even without any actual measurement program, companies can measure something of the process, e.g. cost, profitability, time, or the product, e.g. size, number of errors, downtime. The measurement programs presented in the literature are usually based on the experiences of large software organisations. Small organisations don't have specialised resources for measurement, and they can have different information needs compared to larger organisations.

SPICE contains a lot of references to measurements. A measurement process is defined, many management practices are related to measurement and some work products are classified as measures.

As part of the BootCheck analysis the companies were asked to think about the most important improvement areas and to prioritise process categories accordingly. In addition they were asked to list the most important processes. The two most important process categories were considered to be Engineering (ENG) and Management (MAN), while Organization (ORG) processes were of least interest. In addition, only one of the companies indicated interest in the ORG.5. Measurement process [7]. This confirms that small organisations have very little interest in measurement and that the use of software metrics in small organisations is very limited.

To stimulate companies' interest in more precise measurement the project started to offer training containing measurement aspects. The first course provided was titled "Software project size estimation" and it introduced the participants to function point analysis. The second course was about "Risk management in software projects" and the third course helped companies to perform "Customer satisfaction surveys". All these courses contained information and practical guidance in different measurement types. The companies expressed lots of interest in these courses and at least some customer satisfaction surveys and risk analyses have already been made.

Basis to develop actual measurement programs will be laid in workshops and

additional courses. The idea is that the companies will co-operate in the workshops to create measurement programs that fit general small company needs as well as the individual needs of those who need the results.
To help measuring the benefits of SPI, Zahran has divided measures in three classes: process-related; project-related; and product- and customer-related measures [10]. Whatever the basis of the measurement program development will be, the steps in the beginning should be small. Small companies can and will not dedicate specialised resources for measurement and their interest towards measurement itself is also very limited [9].

Training activities

Training is targeted to support process implementation by updating personnel skills and knowledge on software engineering methods and technologies. The project has offered a wide variety of courses and seminars in the area of software engineering and management. Training services has mainly been provided by subcontractors and partly by the project itself. In the near future some of the courses will be repeated and new topics will be included. The courses have been categorised according to SPICE process categories as follows:

Organisation
- Introduction to software processes
- Software process improvement models
- SPICE in assessment and improvement
- Process definition and design
- SPICE assessor training
- Product focused SPI
- Software engineering measures and measurement
- SPI update seminar
- Management of SPI
- Team work

Management
- Project management in information technology projects
- Risk management in software projects
- Software project size estimation
- Management program
- Quality management
- ISO9000 for software SMEs
- Software engineering process
- Software project management

Customer-Supplier
- Requirements management in software projects
- Customer satisfaction surveys
- Help desk working

Engineering
- Introduction to relational database design
- TCP/IP structures and concepts
- Seminar on objects

Relational database design and implementation
Relational database design with CASE tools
Introduction to SA method
Data structures and algorithms
Software testing
Object oriented specification and design
Java-based application engineering environments

Support

Webmaster
Process automation and documentation in software engineering
Documentation and configuration management tools
Inspections and reviews in software engineering
Software documentation
Online help design

The initial goal was to arrange training for 100 persons during the project but already the project has clearly exceeded that. Main reasons for the extensive interest is that many courses are tailored according to company needs and most courses are held close to the companies.

Future of the project

Next the follow-up process assessments will be carried out to confirm the SPI achievements. Special attention will be paid to encourage continuous self-assessments in the companies. The consultation and the training activities will continue with the pilot-companies.

The second phase of the project will ensure the continuity of the SPIN and the active co-operation within the Satakunta software industry and enlarge the number of the companies participating the *SataSPIN* project. An essential part will be to create an SME-oriented SPI handbook, which will include recommendations for the SPI priority determination and the applicable process models.

Pori School of Technology and Economics will continue to strengthen its expertise in SPI and thus serve the software industry.

References

[1] BOOTSTRAP Institute: BootCheck assessment tool, http://bootstrap.ccc.fi.

[2] ISO/IEC TR 15504-2:1998 Information technology - Software process assessment - Part 2: A reference model for processes and process capability.

[3] ISO/IEC TR 15504-5:1998 Information technology - Software process assessment - Part 5: An assessment model and indicator guidance.

[4] ISO/IEC TR 15504-7:1998 Information technology - Software process assessment - Part 7: Guide for use in process improvement.

[5] Paulk M., Curtis B., Chrissis M. B. & Weber C., Capability Maturity Model for Software, Version 1.1. *Technical Report CMU/SEI-93-TR-24,* SEI 1993

[6] Pressman, Roger S., Software Engineering: a practitioner's approach, McGraw-Hill 1992.

[7] Varkoi, Timo, Mäkinen, Timo & Jaakkola, Hannu, Process Improvement Priorities in Small Software Companies, *Proceedings of the PICMET´99*, Portland, Oregon, 1999

[8] Varkoi, Timo & Mäkinen, Timo, Requirements Traceability Improvement in Small Software Organizations, *Proceedings of the PNSQC´99*, Portland, Oregon, 1999 (in press)

[9] Varkoi, Timo, Development of Measurement Programs to Support Software Process Improvement in Small Software Companies, *Proceedings of the FESMA´99*, Amsterdam, 1999 (in press)

[10] Zahran S., Software Process Improvement: Practical Guidelines for business success, Addison-Wesley, 1998

Mid-term results of the SPIRAL Network Development

Béatrix BARAFORT, Anne HENDRICK

*Joint contribution of the SPIRAL*NET[1] team*

Centre de Recherche Public Henri Tudor, L-1359 Luxembourg

Introduction

The European Commission (through ESSI - European Systems and Software Initiative) supports Software Best Practice Networks (in other words ESBNET projects) in order to disseminate best practices.

SPIRAL*NET is an ESBNET project[1]. It focuses on **best practices in the Customer/Supplier processes quality management** (CSPQM) and aims at **optimising and generalising** such **practices** in a French speaking area composed of the Grand-Duchy of Luxembourg, Wallonie (the French speaking part of Belgium) and the French Lorraine. The Customer/Supplier processes quality management covers Customer/Supplier processes, support processes such as quality assurance, joint reviews, configuration management, documentation management, project management.

The objectives of the SPIRAL*NET project are:

- to make the market aware of best practices on Customer/Supplier processes and associated support processes,
- to improve the visibility and the access to structured information related to the regional software market,
- to provide and standardise the market with CSPQM tools selected from best practices and adapted to regional practices,
- to support CSPQM implementation with the shared tools,
- to provide the market with service offers on CSPQM (services supported by the project partners and a qualification program proposed by the project).

The partners of the project are the Centre de Recherche Public Henri Tudor in Luxembourg, as co-ordinator, the Centre de Transfert de Technologie de Charleroi (innovation Centre created from the University of Namur – Facultés Uuniversitaires Notre-Dame de la Paix) in Belgium and the "Unité de Formation Recherche – Mathématiques et Informatique" of the University of Nancy 2 in France. They developed an implementation strategy throughout 5 layers in order to meet the objectives :

- to heighten the market awareness,
- to set up a common electronic platform,
- to select and to share support tools,
- to support the implementation of CSPQM and the use of common tools,
- to provide the market with qualification and certification.

Following this 5-layer strategy, this paper describes how significant results have been reached after one year of project running[2]. Then, before the conclusion, the dissemination

[1] SPIRAL*NET is the ESSI ESBNET project 27884

actions show how the SPIRAL*NET project and other European projects results are spread, and how European co-operation is established in order to develop the network.

To heighten the market awareness

The software market has to be made sensitive to the benefits of a formalised approach and of Customer/Supplier quality processes through standardised tools and dedicated support services. In order to do so and to promote the SPIRAL network development, several actions had been organised like awareness events and workshops, participation in local trade fairs and exhibitions, and prospective visits.

The first semester of the SPIRAL*NET project (June to December 1998) was stand out by preparation activities in order to enable the implementation of the electronic platform infrastructure with basic facilities, and the definition of products and services to be launched in 1999: basic Extranet services and the regional software market directory, SPIRAL activities such as working groups and support services to propose to SSDs and SMEs. All along the first semester of the project, prospective visits were planned and companies were called on. The objective was to aware them on CSPQM, to attract companies in the network, to take part in activities and more particularly to include them in the software regional directory. A co-operation agreement has been developed and companies are asked to sign it in order to register and formalise their membership in the network.

At the beginning of 1999, a new phase started and was named the "call stage" or network launching for implementing the first set of services and activities. The SPIRAL network activities were promoted and the following actions and/or activities were proposed:

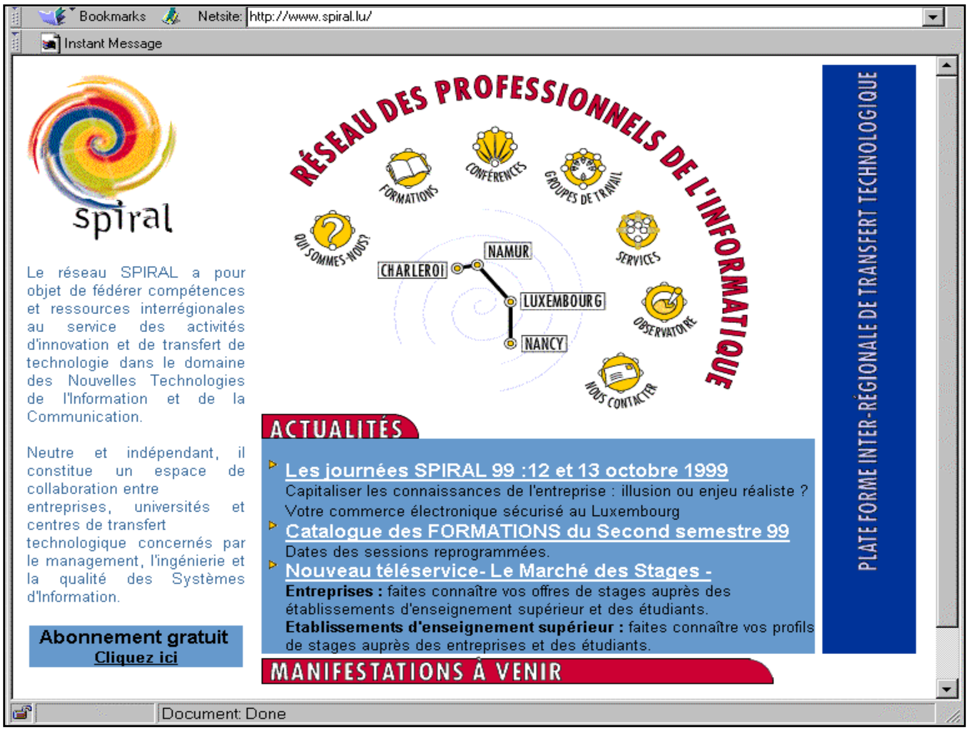
- to register and formalise the membership in the network,
- to appoint a quality interlocutor (from the SPIRAL*NET project team) for the company,
- to register and provide descriptive information about the company's IT activities and competencies in order for the company to be listed in the regional software directory,
- to make the company know the training courses available in the SPIRAL training catalogue,
- to participate in working groups,
- to be assisted and advised in Software Process Improvement (SPI) (SPI services are proposed such as micro-assessments, SPICE assessments, improvement plan determination, SPI actions implementation, customer/supplier relationships assistance),
- to participate in a specialised training cycle in SPI.

To set up a common electronic platform

In order to attract as many IT actors as possible in the regional software market, a common electronic platform has been prepared at the very beginning of the project, with basic facilities. An electronic frame has been set up in order to support all activities related to the SPIRAL network. Several entry points are provided through icons for:

- presenting SPIRAL partners,
- accessing the training catalogue where CSPQM related courses are proposed,

- announcing local, regional and international conferences,
- presenting actual and future working groups,
- proposing electronic services like a directory of the regional software market, a search engine on IT quality standards which are available in the Centre Henri Tudor's library and a work placement market service, and CSPQM services (support services and mentoring) that can be performed by the SPIRAL*NET team.



Le réseau SPIRAL a pour objet de fédérer compétences et ressources interrégionales au service des activités d'innovation et de transfert de technologie dans le domaine des Nouvelles Technologies de l'Information et de la Communication.

Neutre et indépendant, il constitue un espace de collaboration entre entreprises, universités et centres de transfert technologique concernés par le management, l'ingénierie et la qualité des Systèmes d'Information.

Abonnement gratuit
Cliquez ici

ACTUALITÉS

- Les journées SPIRAL 99 :12 et 13 octobre 1999
  Capitaliser les connaissances de l'entreprise : illusion ou enjeu réaliste ?
  Votre commerce électronique sécurisé au Luxembourg
- Catalogue des FORMATIONS du Second semestre 99
  Dates des sessions reprogrammées.
- Nouveau téléservice- Le Marché des Stages -
  Entreprises : faites connaître vos offres de stages auprès des établissements d'enseignement supérieur et des étudiants.
  Etablissements d'enseignement supérieur : faites connaître vos profils de stages auprès des entreprises et des étudiants.

MANIFESTATIONS À VENIR

Figure BB&AH.1 : Homepage of the SPIRAL web platform

The directory of the regional software market

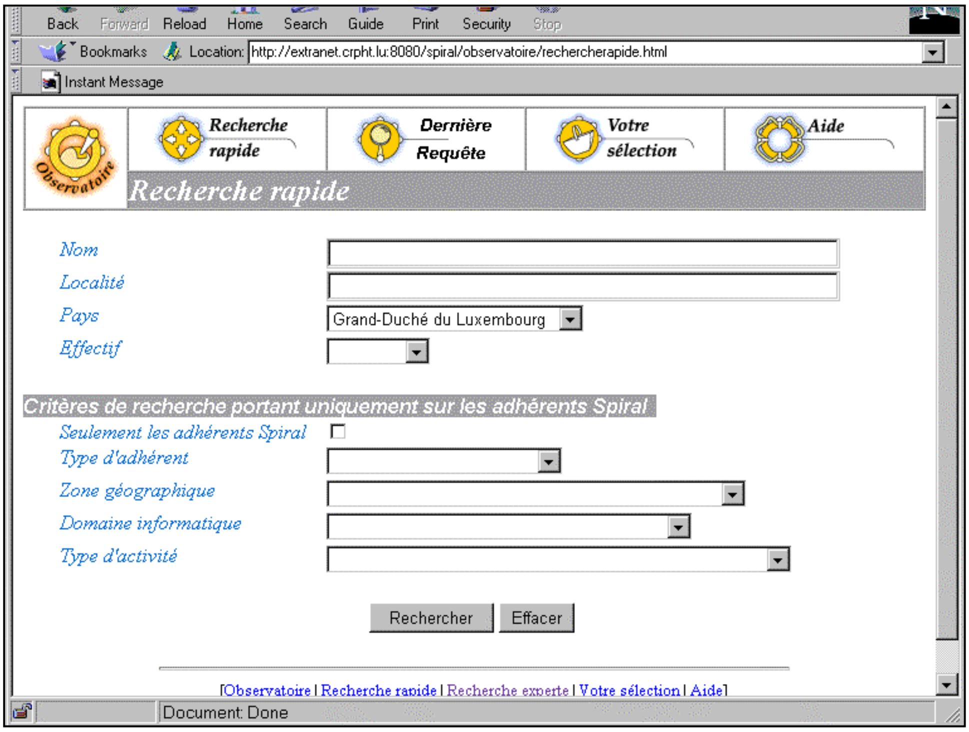
The main component of the electronic platform is the directory of the regional software market. This directory is aiming at providing structured information on IT suppliers and customers. So that, for instance, buyers are able to electronically consult on their needs. The directory shows IT services and competencies of the Lorraine, Wallonie, and Grand Duchy of Luxembourg.

The displayed professionals are IT departments, software houses, IT independent workers, associations and individual members. The directory is intended to people seeking representative and precise information on one or more actors of the software market and competencies or services meeting their needs.

After a first development phase, a prototype has been put online since the beginning of January 1999 (with a bunch of a dozen SPIRAL member companies and a hundred of regional companies copied from a public file, with limited information such as descriptive and activities). It is accessible to general public through the SPIRAL Web site. In its current version, it aims at gathering information on the companies of the SPIRAL network targeted area. These data are primarily their descriptive information, their activities and references. This service allows the search for companies via a multiple criteria search engine. Then it enables the information consultation and retrieval of the search results.

The integration of the companies' competencies was the next goal to reach in the project in order to improve the visibility and the access to structured information related to the regional software market. So, a second phase of the directory development (second semester of the SPIRAL*NET project) has been devoted to the analysis of the context (definition of the IT competencies structure, needs analysis, alternative solutions study, final solution choice) and the implementation (design, tests of the tools, technical implementation). At the end of the summer 99, the new version of the directory will integrate the competencies dimension.

You will find below an example of scenario that can be followed (with user interfaces) by somebody who is looking for companies with specific competencies.



Location: http://extranet.crpht.lu:8080/spiral/observatoire/rechercherapide.html

Recherche rapide

Nom	
Localité	
Pays	Grand-Duché du Luxembourg
Effectif	

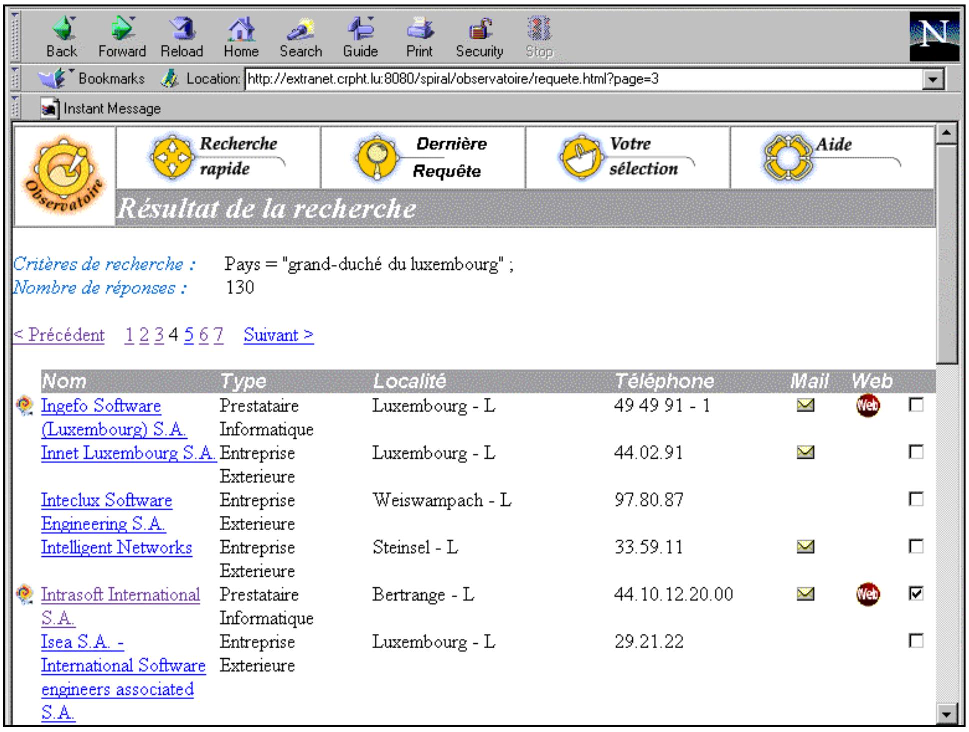
Critères de recherche portant uniquement sur les adhérents Spiral

Seulement les adhérents Spiral	□
Type d'adhérent	
Zone géographique	
Domaine informatique	
Type d'activité	

[Observatoire | Recherche rapide | Recherche experte | Votre sélection | Aide]

Document: Done

Figure BB&AH.2 : Search criteria in the directory



http://extranet.crpht.lu:8080/spiral/observatoire/requete.html?page=3

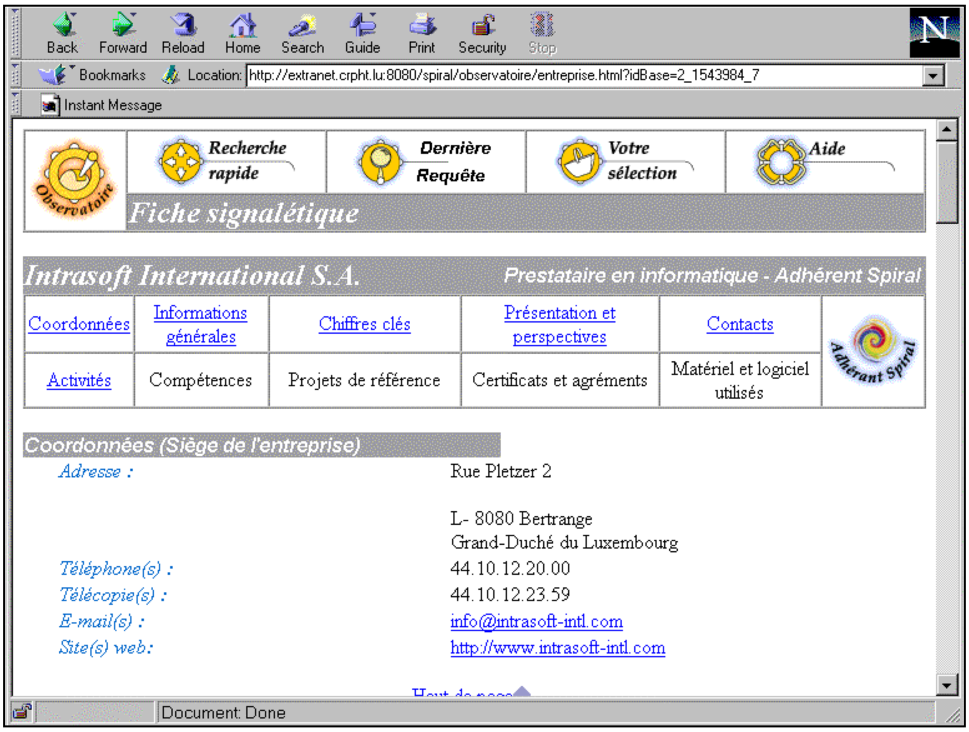
Résultat de la recherche

Critères de recherche : Pays = "grand-duché du luxembourg" ;
Nombre de réponses : 130

< Précédent 1 2 3 4 5 6 7 Suivant >

Nom	Type	Localité	Téléphone	Mail	Web	
Ingefo Software (Luxembourg) S.A.	Prestataire Informatique	Luxembourg - L	49 49 91 - 1	✉	Web	□
Innet Luxembourg S.A.	Entreprise Exterieure	Luxembourg - L	44.02.91	✉		□
Inteclux Software Engineering S.A.	Entreprise Exterieure	Weiswampach - L	97.80.87			□
Intelligent Networks	Entreprise Exterieure	Steinsel - L	33.59.11	✉		□
Intrasoft International S.A.	Prestataire Informatique	Bertrange - L	44.10.12.20.00	✉	Web	☑
Isea S.A. - International Software engineers associated S.A.	Entreprise Exterieure	Luxembourg - L	29.21.22			□

Figure BB&AH.3 : Search results



http://extranet.crpht.lu:8080/spiral/observatoire/entreprise.html?idBase=2_1543984_7

Fiche signalétique

Intrasoft International S.A. — Prestataire en informatique - Adhérent Spiral

Coordonnées	Informations générales	Chiffres clés	Présentation et perspectives	Contacts
Activités	Compétences	Projets de référence	Certificats et agréments	Matériel et logiciel utilisés

Coordonnées (Siège de l'entreprise)

Adresse :	Rue Pletzer 2 L- 8080 Bertrange Grand-Duché du Luxembourg
Téléphone(s) :	44.10.12.20.00
Télécopie(s) :	44.10.12.23.59
E-mail(s) :	info@intrasoft-intl.com
Site(s) web:	http://www.intrasoft-intl.com

Haut de page

Document: Done

Figure BB&AH.4 : Example of a company description

The work placement market service

The market of the training periods is another telematics project whose finality is the implementation of an on-line service on the SPIRAL Web site. It can offer three function levels:

- a c**ompanies** function, thanks to which companies can publish training period offers,
- a **higher education establishments** function, thanks to which a profile for training periods can be presented,
- a **students** function, thanks to which students can post their candidatures.

For each function, it will be possible to consult training period's offers and demands.

This service will be in relation with the directory of the regional software market previously described. Thus, when a company proposing a training period is recorded in the directory database, the training periods market application will automatically recover the data. In the future, this service could be spread out to other fields than the IT one.

The work placement market service will be online at the end of the summer 99.

The telematics forum

A tool named AltaVista Forum 98, accessible by a Web browser complements this electronic platform. The AltaVista Forum is an application that provides an easy way to communicate and to share resources with different groups of people. You can post notes and replies in an online conversation, notify users through electronic mail when new information has been added, share documents, post Internet web site addresses.

The SPIRAL*NET team uses the telematics forum for internal project needs. But the forum is particularly used within the SPIRAL network for working groups and training cycles.

To select and to share support tools

Main topics structuring the SPIRAL*NET activities development

In order to provide and standardise the market with CSPQM tools, the SPIRAL*NET project partners identified main topics throughout the SPIRAL network. These are the following with their associated objectives:

- <u>Project Management</u>
  - to assess IS project management practices and to define a grid for analysing IS projects,
  - to define the project management process needs in terms of tools, organisational structure of methodological framework,
  - to establish the implementation approach of the project management process in order to manage it, to supply it with tools and to improve it,
  - to think about the associated processes and connected activities implementation such as risk management, practices standardisation, results and knowledge capitalisation, indicators set up, competencies management…
- <u>Customer/Supplier relationships</u>
  - to identify and formalise best practices in the requirements engineering field,
  - to define practices to adopt in order to structure customer/supplier relationships,
  - to establish a typical approach for purchasing IT products (hardware, software or service),
  - to examine juridical aspects for customer/supplier relationships.
- <u>Software Engineering</u>
  - to introduce software engineering and support activities topics all along the development lifecycle of any IS projects, by more particularly focusing on

requirements elicitation and requirements analysis, design, quality assurance, documentation management, configuration management and with presentation of methods, case studies, testimonials…

- to provide tangible results in the form of synthesis, templates, checklists, a glossary…

A technology watch is made on these topics. All elements related to them are considered in order to contribute to the preparation of SPIRAL activities and to provide the electronic platform with quality tools such as software, case studies, framework models, questionnaires, reference sets, and recommendations. A Project Management tool has been particularly studied for this period and is experimented in the SPIRAL*NET project itself.

All the approach is supported by a methodological set stemmed from *the "Process Professional Portfolio*"[3] licensed by Compita Ltd (UK). This methodological framework is associated with training courses. The whole project team has been trained to Process Professional Assessment in the context of the ISO/IEC 15504 standard[4] (also known as SPICE: Software Process Assessment and Capability dEtermination), to "Supplier Management" and "Assessing Suppliers".

Working groups objectives

It is through working groups focusing on the three previously mentioned topics that recommendations and harmonisation means for best practices are going to be established.

The working groups are a meeting point and an experiment exchange space between IT actors willing to improve their professional practices and to contribute to the IT profession enhancement. These meetings gathering peers focus on topics related to management and Information System (IS) engineering. Participants exchange viewpoints, share their analysis and experiences, invite experts and thus participate in the development and the animation of the SPIRAL network.

By gathering fellow workers from several activity sectors, the working group is representative of the IT profession needs. Its works aim at formalising recommendations, synthesis and studies development in the covered field, and then producing deliverables. The working group arouses harmonisation efforts of the practices and also participates in the writing of a professional charter.

As it was quoted before, the three organised working groups focus on the improvement of software engineering practices, project management practices and customer/supplier/ relationships. Each working group sets its goals and expected deliverables. The outcomes are disseminated throughout the SPIRAL network in various forms such as recommendations and guidelines, templates and user guides, documentary and methodological frameworks, and market studies. These works are enhanced by testimonials and case studies presented by working groups' members or external participants.

Working groups organisation and running

At the end of March 99, three free working groups have been launched via a round-table conference. Until the 99-summer start, two sessions of each working group occurred.

Every working group meets monthly. Between each session, participants can access the telematics forum in order to consult and supply a document base, to exchange and summarise the main questions, answers and experiences.

<u>Working groups inter-session communication</u>

The telematics forum. is a useful communication mean between the meetings for posting documents, announcing events and next meetings agenda, asking and answering questions… After 3 months of working groups running, the forum is mainly used for consultation. The working groups members watch meeting minutes and handouts and the next sessions' programme. One of the challenges of the forum use consists in a more sustained involvement from the working groups' participants as far as the groups orientations are concerned and in the expression of discussion topics on the studied themes. This objective could be reach when the members are involved for producing deliverables.

Working groups leading

An expert in the domain leads each working group. Leading tasks concern:

- technological watch in the topics covered by the working group: bibliography, Internet survey, searches for similar experiences within the SPIRAL network…
- next session preparation: scientific content of the sessions, search for speakers, discussion topics and preparation of the deliverables to produce,
- communication and spreading liven up throughout the electronic forum.

Working group leaders have quite a tricky position in the group running. They have to combine an expert attitude (to keep the debates prolific) with a leader one in order to encourage participants to voice their opinions and experiences. This double role means a balance to reach; this takes some time before the leader knows the group and vice versa. Major changes have already been noticed between the first session and the second one wherefore exchanges between participants increased.

Meetings running

Most of the time meetings are organised with the following outline: theoretical presentations, case studies presentation, debate/discussion, thoughts about the deliverable to produce on the studied topic. Theoretical presentations and case studies testimonials are the means to transfer information to participants who directly benefit from them. They trigger discussions about current practices within the participants companies. These discussions are rather rich even though not structured. And because of lack of time, deliverables thoughts rarely succeeded.

After three months running, the statement is that the meeting agendas were often too ambitious. So the deliverable thoughts that were planned at the end of the session could not be performed. The working groups' success can be measured through the participants' assiduity and the produced deliverables. For the first sessions, the objective was to mainly win the "loyalty" of the registered participants. Then the sessions were more focused on theoretical presentations than on deliverables production. In order to reach the deliverable production goal, the session running will have to be tailored; sharing of experiences, participants' involvement in the working group running and exchanges between them will have to be highly stimulated.

The second half of the project will take particular care to develop the working groups and to deduce regional software best practices (in the form of a directory associated to an improvement best practices guide) from the working groups' outcomes.

To support the implementation of CSPQM and the use of common tools

The implementation consists in providing the network with a direct support in launching CSQPM and their associated tools in business relationships.

Training

The training catalogue is the showcase of available training courses in the SPIRAL network. The SPIRAL*NET team particularly contributes to three developed topics among the following channels: Information System Project Management, Software Practices Assessment and Improvement, Improvement of Customer/Supplier Relationships.

Support services and mentoring

Support services provided to companies throughout mentoring projects have been performed for the first half of the project. Here are the main features of the types of services that were proposed:

- Assistance in the elaboration of a management plan for the IT infrastructure
- Assistance in defining an IT strategy
- Assistance in the IT architecture design based on new technologies
- Coaching and advising for the re-engineering of the information system
- Coaching and advising for supplier selection
- Coaching and advising for supplier follow up during the solution implementation
- Software quality standards awareness meetings
- Improvement quality project definition in an ISO 9000 certification preparation context
- Assistance in project Management and quality assurance for project activities

All mentoring actions and support services are SPI experiences and CSPQM implementations in the SPIRAL network. They are progressively formalised in order to contribute to a case studies corpus. Each case study adopts the same structure: introduction, context of the firm, origin of the support service, description of the service (objectives, actors, project running), firm results, outcomes/lessons learned/capitalisation for the SPIRAL network, plan for the future.

Micro-assessment

A micro-assessment service has been developed and performed within several companies. A company contacted within the SPIRAL network can be put through the questionnaire by telephone. It addresses the vision of the company and the CSPQM practices. Results of the questionnaire point out the planned improvement actions and their effects in the company. The micro-assessment service is described with more details below.

In the assessment context, regular contacts and sharing of experiences have been established with a project team, which works for the Walloon region[5] on the building of an SME dedicated framework for software assessment (via a project named OWPL for "Observatoire Wallon des Pratiques Logicielles[6]"). Common work has been accomplished for the micro-assessment development.

The aim of the micro-assessment is to give a first outlook of the software practices in an organisation, to make a diagnosis and guide the next steps of software process improvement. The main requirement that drives the design of this model is to be as less costly as possible, in time and money.

So, the designed model corresponds to a half an hour interview based on a well-prepared questionnaire. The questionnaire covers six key lines selected as the most pertinent and the most prior to target organisations. The questionnaire has been based on SPICE and CMM concepts. All adaptations were made by considering that the micro-assessment had to be short in time and money, and had to particularly suit SSDs and SMEs. So the key lines are the following:

- quality assurance,
- customers management,
- subcontractors management,
- project management,

- product management,
- training and human resources management.

The questionnaire includes a dozen of questions covering the above mentioned topics. Questions are open, and each of them is associated with one or more sub-questions allowing the interviewer, if needed, to adjust and refine the information he gets. Micro-assessments are performed by a member of the SPIRAL*NET team; the interviewee has to be in charge of software quality in the organisation; this is usually one of the executive staff members or the quality engineer, if this function exists.

Answers are interpreted according to a fixed rating scale. The results are presented graphically. Figure 3 below gives an example of the resulted grids. The first grid is the detailed evaluation results according to the selected practices while the second one is a summarised picture according to the six selected key lines.

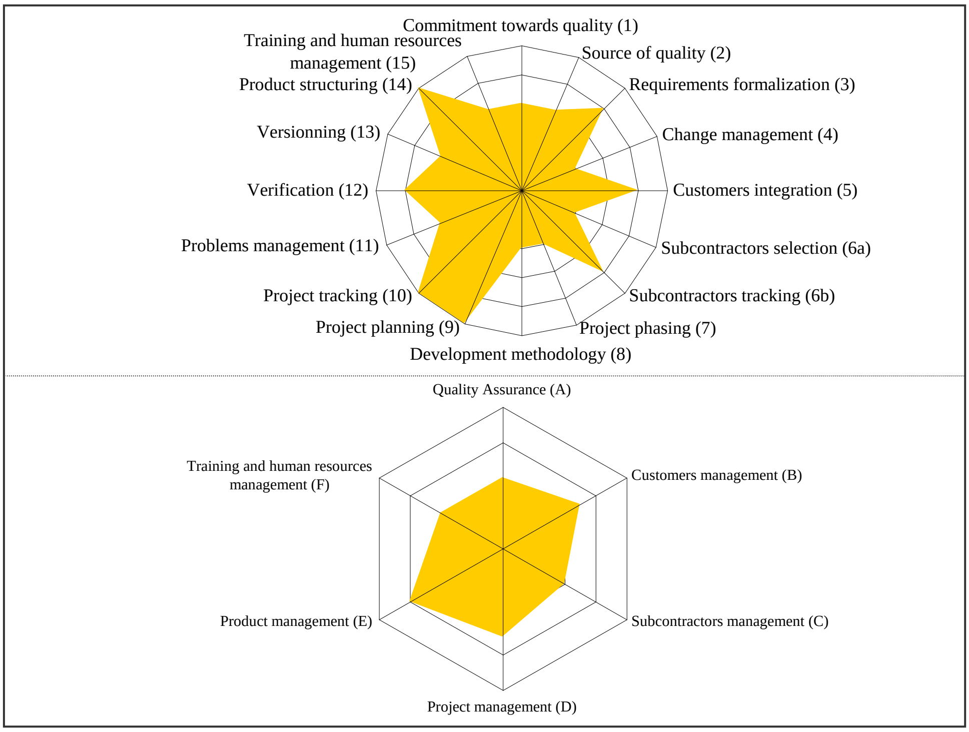


Figure BB&AH.5 : example of micro-assessment resulted grids

The results of the micro-assessment are drawn up in a report that first briefly presents the approach, then develops the results of the questionnaire and summarises them according to the six key lines, analyses them according to the situation of the assessed organisation (the age, the history, the declared goals…) and finally gives some recommendations.

The micro-assessment has been performed with about 20 representative organisations (IT small companies, IT services in other businesses, public administrations using IT). The experience showed that the micro-assessment model is very attractive for SSDs and SMES as a tool to start with SPI, mainly because of its extreme simplicity. All of the assessed organisations declared to be happy with the results and want to carry on their SPI efforts. Up to now there are no significant trends in the regional software practices. Nevertheless, the micro-assessment will enable to measure the impact of the SPIRAL*NET project on the organisations and to show the maturity level enhancement after implementing SPI actions (the micro-assessment is proposed every six months to the companies).

To provide the market with qualification and certification

In order to ensure continuity of the SPIRAL proposed services, the network will develop a qualification and certification process. In that context, a SPI dedicated training cycle has been defined. It is named "Amélioration des Pratiques Logicielles" (APL) for improving software practices.

Training cycle context

This training cycle aims at producing quality engineers in the Information System (IS) field. At the end of the cycle, attendees will be able to implement a software quality process in their company after having:

- initiating a software practice improvement programme and successfully managing pilot actions in a business key area of their company,
- deploying and implementing quality assurance activities in IS projects,
- mastering a methodological baseline and competencies in order to combine business, organisation and the company's strategy with its improvement goals.

The training cycle addresses anyone involved in a quality process in an IT department or a software house.

The cycle alternates theoretical sessions, case studies and experiments analysis with practical actions within each attendee’s respective company. An on-site monitoring is proposed. The unifying thread of the cycle is the software practice improvement project adapted to the company’s goals and specificity. The dynamics induced by the control of actual quality actions enables the training cycle participants to gain concrete experiences. The cycle progress is organised with 10 monthly sessions interspersed with on-site assistance dispensed by a SPIRAL*NET team member.

Each participant have access to the telematics forum which is organised around studied topics in order to consult and supply a document database, to exchange and capitalise questions, answers and experiences from participants and trainers.

Training cycle structure

The SPIRAL*NET SPI training cycle has been built according to a progressive approach, based on the methodological framework *“Process Professional Portfolio*”[3] of Compita Ltd (UK). This methodological portfolio enables to structure the SPI approach and includes software process assessment, improvement and capability determination concepts. The Compita's framework is made up of a process model that is ISO/IEC 15504[4] compliant. Other frameworks will help to structure the approach. These are for instance the ISO 9000[7] standards, or other assessment and improvement frameworks such as xx-CMM (Software-CMM[8], People-CMM,…).

Guidelines of the ISO/IEC 15504 standard[4] recommend 8 steps in the "Guide for use in process improvement" (Part 7). The adopted approach for the APL training cycle is based on these SPI steps, gathered in 4 main stages. For each of these stages, one or several training sessions are the opportunity to tackle associated concepts, and to explain appropriated tools and techniques. This is aiming at progressively building the methodological framework that is the main outcome of the cycle and to give the participants all necessary components in order to implement improvement actions within their company, all along the training sessions. The outlines of the APL training cycle programme are the following:

Initialisation / Launching

- *Kick-off of the improvement approach*
- *Discovery of the SPI context and introduction to a process oriented quality approach*
- *Quality standards and more specifically software standards (ISO/IEC 15504, xx-CMM, ISO 9000)*
- *Key processes selection approach*

Definition / Organisation

- *Implementation of a process improvement environment in a company*
- *Diagnosis, assessment, audit of software practices*
- *Improvement plan definition*

Implementation

- *SPI actors*
- *Activities related to the process performance (base practices)*
- *Activities related to process management (to plan and control process practices, quality assurance activities)*
- *Activities related to process improvement (to control and measure processes, the technological and human resources environment, to formalise processes)*

Improvement

- *The continuous improvement cycle (experiment, adjustment, institutionalisation)*
- *The capitalisation of experiences*

Figure BB&AH.6 : Structure of the APL training cycle

The inter session

All along the training cycle, participants will have the opportunity to take benefit from assistance days included in the APL package. This monitoring will be provided by SPIRAL*NET team members. They will act as a mentor within each company. The participants have to plan key improvement actions and choose the SPIRAL*NET on-site assistance which suit them best among activities like:

- mini-assessment (for instance, a SPICE assessment of a single process),
- deliverable and project reviews,
- specific assistance on project and/or process
- animation / participation in working groups in order to formalise processes,
- templates, procedures, framework development,
- tools specifications, quality service specification, …
- normative and technological watch.

The telematics forum also plays a part between each session. It allows participants to share documents and points of view related to the studied topics.

APL training cycle progress

The APL training cycle definition phase has mainly occurred for the second semester of the project. At the same time, the promotion of the cycle was performed in order to attract potential quality engineers. The project target was set to 6 quality engineers to be trained. Up to now, 5 registrations have been recorded. The training sessions will occur during the second half of the project (from July 1999 to June 2000).

Dissemination actions

Dissemination is dealing with the promotion of the network and wide dissemination of the SPIRAL*NET results throughout all regions of Europe. To do so, several dissemination actions happen all along the project.

Presentations were made by SPIRAL*NET team members in local events such as the annual SPIRAL 98 conferences and in international conferences such as EuroSPI'98, SPI'98, EuroSPI'99.

European exchanges and co-operation are also developed : dissemination materials were provided by European projects (SPIRE[9] – ESSI Project 23873 and SMILE[10] – ESSI Project 23973) and distributed throughout SPIRAL*NET activities such as training courses, working groups, awareness events,… In the same context, the French FESPINODE representative organised two events in Luxembourg and Belgium and took part in both round-table conferences that occurred for launching working groups.

All other types of co-operation and exchanges are welcome. It can be as various as inviting CSPQM experts to participate in local and regional events or providing software facilities via the electronic platform. All the means that can favour the SPIRAL network development are encouraged. All opportunities for the SPIRAL network model to be instantiated again and/or to be extended in any way are studied.

Conclusion

After one year of project running, the SPIRAL*NET results are globally satisfying in terms of awareness impact and welcome of the initiative. A great interest has been recorded for the SPIRAL network in connection with the harmonisation of the market software practices, with the acknowledgement of SPI initiatives, approaches, and competencies in the regional companies aiming at improving their practices. Moreover, this interest has been shown despite a strong mobilisation of human resources in the companies for Euro and Y2K projects.

The second half of the project will consist in implementing all prepared activities and in gathering as many actors as possible within the network in order to study label aspects, acknowledgements mechanisms by the IT professionals themselves, and the maintenance and means to perpetuate SPIRAL*NET activities after the end of the project.

References

[1] SPIRAL*NET Project Programme - ESSI Project 27884 - V 2.0, May 1998

[2] SPIRAL*NET Mid Term Report - ESSI Project 27884 - V 1.0, June 1999

[3] Compita Ltd, Process Professional Process Portfolio, Process Professional Library Services, 1996

[4] XP ISO/CEI TR 15504, Parties 1, 2, 3, 4, 6, 7, 8, et 9 : Septembre 1998 et Partie 5 : Décembre 1998, Technologies de l'information - Evaluation de processus de logiciel, Edition AFNOR en français du référentiel ISO/SPICE

[5] Habra N., Du Bois P., La crise du logiciel : vers une démarche d'amélioration des processus logiciels dans les PME Wallonnes, in : *revue Athena de la DGTRE du Ministère de la Région Wallonne,* Namur, B, September 1998

[6] Habra N., Niyitugabira E., Lamblin A-C., Renault A., Software Process Improvement in Small Organizations Using Gradual Evaluation Schema, In Proceedings of the International Conference Product Focused Software Process Improvement, PROFES'99, Oulu, Finland, 1999

[7] EN ISO 9001, Systèmes qualité - Modèle pour l'assurance de la qualité en conception, développement, production, installation et prestations associées, CEN, Bruxelles, B, Juillet 1994

[8] Dymond K.M., Le guide du CMM (SM) - Introduction au modèle de maturité CMM, Traduction : A.Combelles et al. - Objectif Technologie, Cépaduès Editions, Toulouse, F, 1997

[9] The SPIRE Handbook : Better, Faster, Cheaper Software Development in Small Organisations - ESSI Project 23873 – 1998

[10] SMILE (Spreading Multimedia Information for Learning and Enlightenment about Software Process Improvement) CD-Rom, Multimedia information system and CD-Rom funded by the ESSI Project 23973 – 1998

We would like to thank all SPIRAL team members who contributed to this paper :

Jean-Charles Bernacconi, Pierre Brimont, Yves Collet, Sandra Grunewald, Véronique Henrotte, Naji Habra, Jean-Pierre Jacquot, Anne-Catherine Lamblin, Jean-Pol Michel, Eustache Niyitugabira, Alain Renault, Jeanine Souquières, Norbert Vidon

CV of the authors

Béatrix BARAFORT

- Co-ordinator of several projects of process assessment (SPICE) and improvement programs.
- Currently project leader of SPIRAL*NET (ESSI Project 27884).
- Qualified assessor (certificate of achievement of "Process Professional Assessment").

Anne HENDRICK

- Member of the board.
- Branch manager for Software Process Quality.
- Co-ordinator for the SPIRAL platform. This resource centre aims to enhance information and experience exchanges between IT professionals. Responsible for the definition of service offers in the domain of software engineering, process improvement and quality management, such as consulting, technological and methodological assistance, specialised training, working groups and forums.
- Qualified assessor (certificate of achievement of "Process Professional Assessment").

Centre de Recherche Public Henri Tudor

The Centre de Recherche Public Henri Tudor, founded in 1987 as a public research centre, was created to promote innovation and technological development in Luxembourg. The Centre's goal is to improve the innovation capabilities of the private and public sectors by providing support services across the main technology-critical areas : information and communication technologies, industrial and environmental technologies. It is assisted in its mission by a diversified network of industrial and institutional partners.

The Centre de Recherche Public Henri Tudor participates in European Union programmes including ESPRIT, Craft, Info 2000, LIFE and Telematics Applications Programme. As a result, Luxembourg businesses are able to draw on the knowledge and expertise of Europe’s greatest research centres.

The Centre is also actively engaged in inter-regional co-operations within the "Grande Région" (Saarland and Rheinland-Pfalz in Germany, Lorraine in France and the province of Luxembourg in Belgium). It is a co-founder of the European College of Technology, a tri-state initiative based in the European Development Pole at the Athus-Longwy-Rodange intersection, and contributes to the innovation programmes of the EU Structural Funds.

Main figures

- a full-time staff of 130
- 5 research laboratories
- 4 innovation support services
- 6 technology resource centres
- annual turnover of more than ECU 6 millions
- 60 % self-funding

Software Engineering in the UK – A Brief Report

Dr Zaigham Mahmood
School Of Mathematics and Computing
The University, Derby, UK

- Introduction

In 1997, a project was undertaken to investigate the state of software engineering in the UK. The objective was, mainly, to gain insight into the way the industry carried out the practice of software development but also to investigate the sources of difficulties, problems and concerns inherent in the process. The underlying purpose was to discover trends and gather information for course designers.

It was decided that a survey would be carried out by sending copies of a detailed questionnaire to a number of IT departments of large organisations in the UK. Target organisations were carefully chosen and the questionnaire was appropriately designed. The organisations chosen were all in the business of producing large-scale software and the questionnaire contained questions relating to the various phases of software development life cycle as well as those concerning project management, estimation of cost and time, team working, documentation, quality and standards.

This paper presents the result of the survey. Our study reveals that major concerns exist in the areas of group working, staff experience, time and cost estimation, CASE tools and documentation, although, it appears that the industry has learnt to cope with the difficulties of the requirements phase, project management and quality control.

We hope the information gathered from the study will prove useful for course designers as well as those working in the industry.

- Design of Questionnaire

Response to postal surveys is often poor: usually because the recipients are too busy to find time for such an activity but often because questionnaires are badly designed. We ensured that our questionnaire was well structured and designed in a way that the recipients would find it relatively easy to complete them.

The questionnaire consisted of three parts. Section 1 asked for certain general details about the organisation such as their size, number of staff in the software development section and break down of computer staff (project managers, programmers, designers, systems analysts, other). Section 2 required details of the kind of software development undertaken by the organisations and the way it was carried out. So, there were questions relating to the following:

- type of software produced
- time spent on a typical project
- team size
- frequency of team meetings
- use of methodologies and CASE tools
- quality and standards.

Section 3 referred to problems and difficulties encountered during the development process. Questions were grouped under the following headings:

- user requirements
- development methodologies
- CASE tools
- documentation
- group working
- project management
- cost and time estimation
- standards and quality.

- Target Organisations

The organisations we chose to study were mostly medium-to-large, although there were also a few very large companies. Questionnaires were sent to their IT departments. Their size ranged from 20 to 1000 computer personnel, although a majority of them had in access of 150 computing staff each. They were all in the business of developing large-scale software and included banks, building societies, insurance companies, service and consumer agencies, car manufacturers/designers, communication industry, superstores, a railway agency, an aerospace agency and several software houses.

Questionnaires were posted to relevant named personnel as we wanted to approach those directly involved in the design, development and management of software systems and projects. One hundred questionnaires were sent: 48 were returned and 35 were used in the final analysis. 21 of the companies involved in our research produced financial software, 5 were in the business of producing educational, research and scientific software, 6 organisations belonged to communication industry and 28 of them produced information systems for manufacturing, distribution, retail and other similar applications. Some of them produced more than one type of software.

Of the 35 organisations, 28 had an average development time of 6-12 months for a typical project whereas 9 companies took more than a year on average to complete a typical application. Since the team size varied from project to project and from company to company, it was difficult to determine the average number of person hours spent on a software product. Nevertheless, looking at the figures given for development time and team sizes, we have no doubt that the organisations involved in the study were, on the whole, medium-to-large.

- Analysis of Questionnaires

Summary of our findings will appear in a later section. Here, we present quantitative analysis of the survey using separate headings for clarity.

- User Requirements

We expected to hear some major problems in this area. But, recognizing the fact that requirements will always change, most organisations allowed such changes but then they negotiated new contracts and agreed new time scales and costs with clients. 70% of companies used internally defined 'change control procedures' which included re-costing. 61% of the companies used prototyping to elicit user requirements. Three organisations mentioned Rapid Applications Development (RAD) and PRINCE as a way to handle changing requirements.

Surprisingly, 3% of the respondents disagreed with the statement that *requirements often change during the course of development* and 20% said that they had no formal procedures to handle such requests. One organisation reported that they *put a freeze on further changes*.

Our survey revealed that 60% organisations consulted users at regular intervals and the remaining 40% consulted clients as and when required.

- Development Methodologies

Those involved in the study used at least one well recognised method for software development: some used up to three depending on the nature of the applications. 61% companies used prototyping to establish user requirements and 45% companies followed evolutionary approach [1] to software development. The Waterfall approach [2] in various guises was used by 51% of the respondents. Only 6% of companies used the Spiral model [3] for software development. Other methods used by companies included internally defined methodologies, RAD and DSDM. No major problems were reported. Refer to Fig. ZM 1.

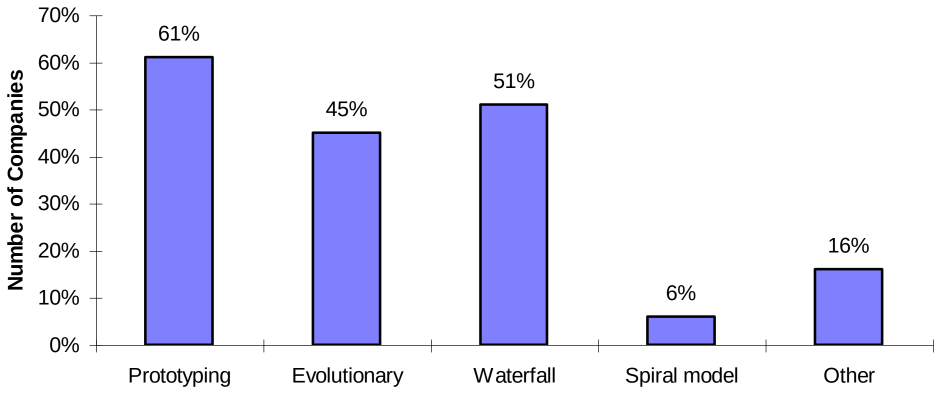


Fig. ZM1: Use of development methodologies

- **CASE Tools**

It was interesting to note that 52% of companies did not use such tools to aid the development process and 6% of companies ignored the question altogether. Of the 42% of companies, who used such tools for software development, used them for various phases of the development process as follows:

• design:	100%
• requirements:	66%
• coding:	66%
• testing:	16%.

Various tools mentioned by our respondents included LBMS, MS Project, PMW, S-Designer, MS Visual Test, Rational Rose, ARIS, JMA, Designer 2000 and Oracle Case. Fig. ZM2 represents the situation graphically.

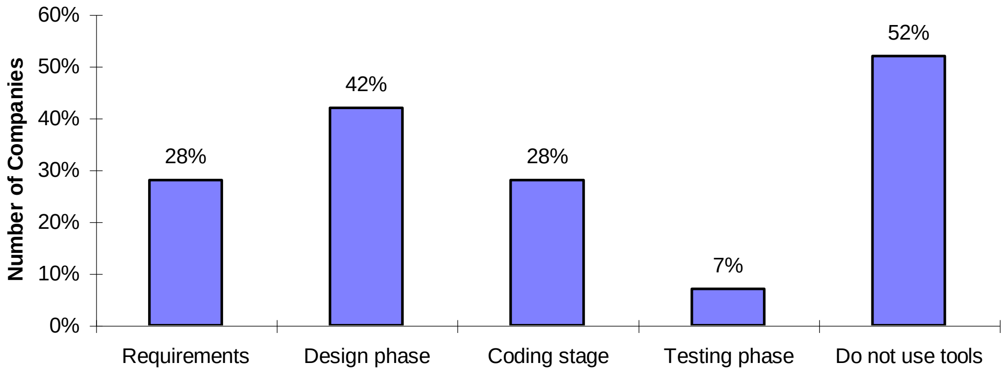


Fig. ZM2: Use of CASE tools

- **Documentation**

Nearly half the respondents reported that their software was *well documented*; the other half were *generally happy* with their documentation. Two companies were *unhappy* about the quality of documentation they produced. Lack of time was cited as the main problem, however, two organisations out of 35 reported no problems of any kind whatsoever.

74% companies told us that they produced documentation at regular intervals and 23% organisations produced it as and when time permitted. Two companies said that they produced documentation only at the end of development. Major difficulties with documentation were cited as follows:

• lack of time:	17 organisations
• dislike/unwillingness:	6 organisations
• keeping pace with changes:	7 organisations

90% of the organisations had formal mechanisms in place to ensure the completeness of their documentation. These included reviews, audits, quality procedures, formal signing off, project plans, management control, walk throughs and checklists. Refer to Fig. ZM3.

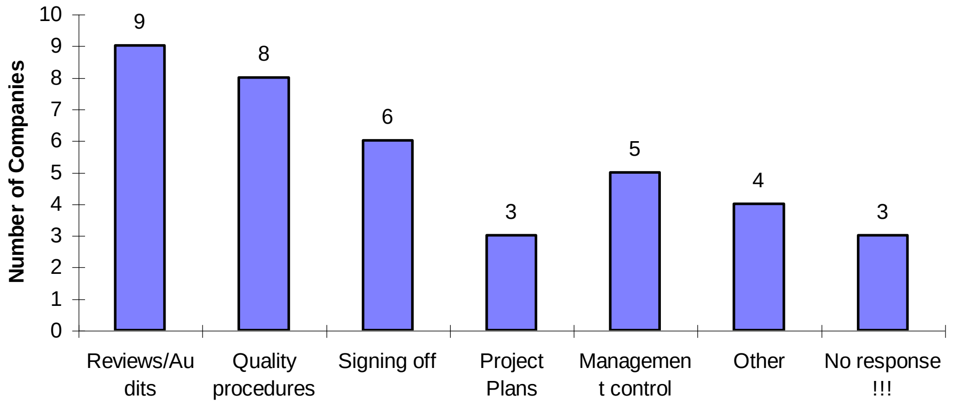


Fig. ZM3: Methods to ensure completeness of documentation

-

-

- Group Working

80% of companies agreed that organisation, co-ordination and monitoring of large teams was generally *very difficult*. Major problems seemed to be *communication between team members* and *inexperienced personnel*. Other problems highlighted by 35 organisations included friction between team members, lack of information and support from senior management, inexperienced team leaders, staff leaving, inappropriate time scales, large groups and communication between teams. Refer to Fig. ZM4. Solutions, not surprisingly, included more training courses, smaller teams and more project reviews.

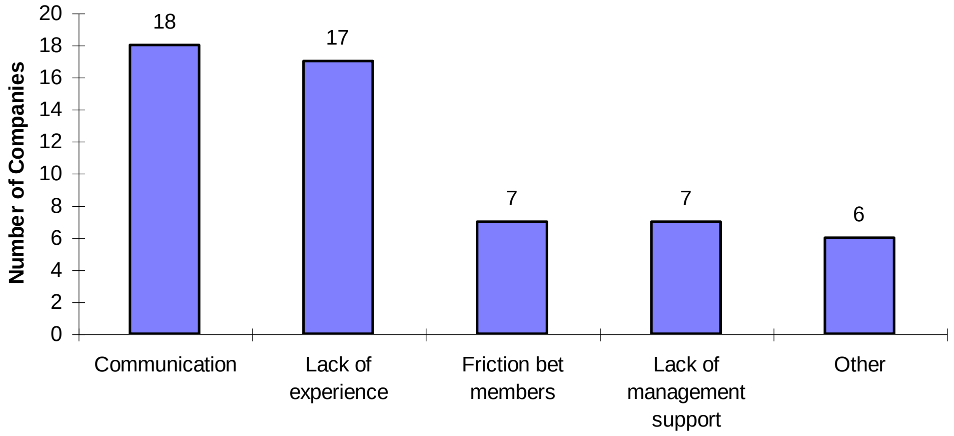


Fig. ZM4: Problems with group working

Two third of the companies held group meetings each day and the rest had regular meetings once a week.

Answering question whether they had any mechanisms in place to resolve team problems, 42% said yes, 12% answered no and, surprisingly, the rest, nearly half, gave no response. 7% organisations admitted that the procedures they have to combat any difficulties were *not very effective*. Several questions from this section of the questionnaire remained answered. It seems therefore that no clear answers exist to several of the usual problems when people work in groups of many. Also, in spite of increasing number of courses offered by institutions of higher education, lack of appropriate skills remains a major problem.

- **Project Management**

Our investigation revealed that MS Project and Project Management Workbench were the most popular tools used for project management - used by 31 organisations. Of the other 4 organisations, two did not use any tools to check the progress of projects and the other two companies used Artemis and Hydra.

No one admitted any weaknesses in the management of projects, personnel or resources though lack of resources seemed to be a major concern.

- **Cost and Time Estimation**

26 organisations (out of 33) did not use tools for the estimation of project times or costs. Three companies used COCOMO [4] and the other four used PMW, Function Point Analysis or internally defined methods. 72% of the organisations told us that they used *past experience* or *careful planning* to estimate the duration of a software project - not a very satisfactory state of affairs.

28 companies admitted that their projects often exceeded the allocated budgets. As shown in Fig. ZM5, our survey revealed that:

- 2 organisations exceeded budgets for less than 10% of projects
- 14 companies exceeded their budget for 10-20% of all projects
- 7 companies exceeded budgets for 20-30% of their projects
- 3 companies overran their budgets for 30-40% of all projects
- 2 organisations exceeded budgets over 40% of the time.

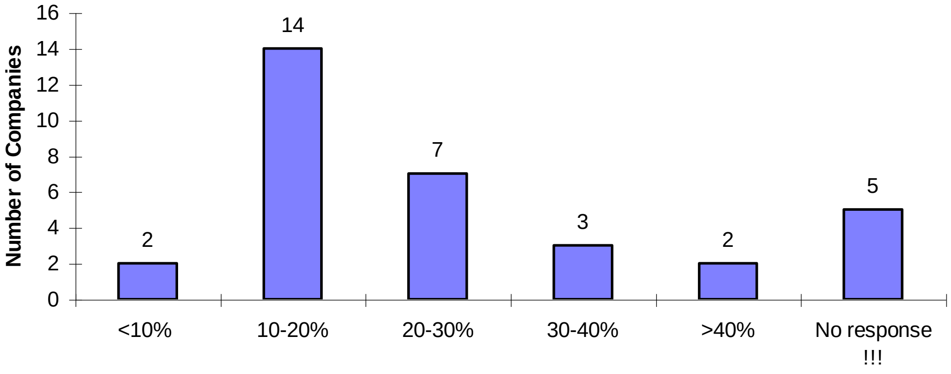


Fig. ZM5: Budget overruns

Only 7 companies delivered their products on time. Of the other 26 organisations, we discovered that:

- 15 companies had 10-30% of their software systems delivered late. Of these:
  - 7 companies exceeded delivery times by about a month on average
  - 5 companies delayed delivery of products by up to 3 months on average
  - 1 company exceeded targets by up to 6 months on average.

- 9 companies had 30-50% of their software products delivered late. Of these:
  - 3 organisations exceeded delivery times by about a month on average
  - 2 company exceeded targets by up to 6 months on average.

- 2 organisations admitted that they delivered more than half of their software late by about 6 months each time.

The above situation is summarized in Fig. ZM6.

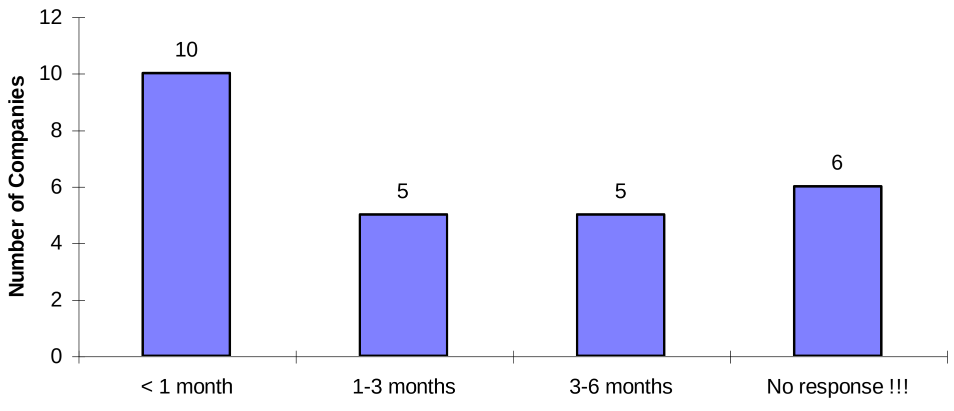


Fig. ZM6: Schedule overruns

Answering the question *what other measures do you take to ensure that projects meet delivery deadlines?*, our respondents provided the following information:

- half of the companies used regular meetings, checks and reviews
- nearly 25% organisations used *careful planning* or *internally defined procedures*
- another 25% reported that they would allocate extra resources such as hiring more staff or getting staff to work overtime
- one company out of 30 said that they would *reduce functionality* to meet delivery targets.

- **Standards**

It was good to know that all organisations had well defined standards and procedures for its staff to follow. 40% of companies used these standards *all the time*, 50% of companies followed standards *most of the time* and the remaining 10% companies followed them only *occasionally*. Nearly half of the respondents regarded such formal standards as *very good* and the other half as *average*.

25% of companies said that there were *too many* standards at their organisations, 6% said there were *too few* and another 6% revealed that their standards were *too complicated* to follow.

- **Quality Assurance**

85% of companies followed recognised quality assurance procedures. Out of a total of 34 companies, BS7570 was followed by 10, Tickit by 14 and ISO 90001-4 by 19 organisations. Some companies were accredited to more than one Quality Standard Schemes. However, surprisingly, 12% of those surveyed relied only on internally defined quality assurance procedures.

For the design and code stages of software development, 80% of the companies surveyed had established *quality assurance groups* within the organisations. However, their staff were not entirely happy with the quality procedures. The mechanisms to ensure that quality was maintained included regular reviews, audits, code inspection and testing. One company said that it was up to the programming staff to ensure that all was well.

- Summary of Findings

Numerous surveys of a similar nature have been conducted in the past 20 years. Our research has shown, yet again, that the process of engineering software is fraught with difficulties despite huge developments in methods, tools and methodologies. Perhaps, *building software will always be difficult* [5]. Our study reveals that major concerns exist in the following areas:

- group working
- staff expertise
- cost estimation
- time estimation
- CASE tools
- documentation.

Main problems in group working is the lack or break of communication between personnel. Nearly half the companies regarded *communication between members of the group* as the major problem. 7% of the companies who have procedures to deal with such difficulties admitted that their methods lack *effectiveness*.

Lack of appropriate knowledge and expertise is another problem highlighted by nearly half of the respondents.

78% of the organisations do not use any tools to estimate time or cost of software projects. Managers rely on their experience to determine time scales, costs and resources required. 28 companies out of 33 admitted that their projects often exceeded the allocated budgets. No clear suggestions were given to resolve the difficulties. Delayed delivery of products is another reality. Only 21% of companies said that they met their delivery deadlines.

CASE tools were used only by 42% of the companies, mainly during design, requirements and coding phases of development. Only 16% of companies used them for testing of software.

Documentation is always produced. However, nearly half of the companies reported that there was often not enough time to produce documentation of sufficiently high quality. Development teams often avoid this activity in favor of more technical work. Keeping documentation up to date is also a major concern.

On a positive note, however, it seems the industry has learnt to cope with the difficulties of the requirements phase, development methods, project management and quality control.

61% of the organisations we surveyed consult their clients on a regular basis and use prototyping as a tool to elicit system requirements. 20% of the organisations, however, lack satisfactory procedures to deal with changing requirements. 45% of those surveyed use evolutionary approach and 51% the Waterfall approach to develop software. The software industry is also beginning to use risk-based approaches to build software and that is encouraging.

MS Project and Project Management Workbench are highly popular tools to control project management, used by 31 out of 35 companies in our survey.

All organisations have good internally defined standards to ensure quality of work. 90% of them use them either *all the time* or *most of the time*. 85% of companies are using well known accredited quality standards which include BS7570, Tickit and ISO 9000 series standards.

- Conclusions

We have presented the result of a survey that was carried out in 1997 to investigate the state of software engineering in the UK. Our survey reveals that managers and team leaders are making good use of management tools for better planning and control and that the industry is being successful in determining what users really require. So, the software industry has made some progress in this regard, however, many of the problems associated with building large systems still remain. Despite development of numerous methodologies, new tools and courses [6] over the years, there is still a need for the following:

- better tools to automate the development process
- appropriate training courses to provide the essential up to date knowledge and expertise including communication and transferable skills
- better estimation techniques so that software products are delivered on time and within allocated budgets.

We hope the information gathered from the study will prove useful for course designers as well as those working in the industry.

- References

[1] Gilb T., Principles of Software Engineering Management, Addison-Wesley, 1988.

[2] Royce W.W., Managing the Development of Large Software Systems, in: *Proceedings of IEEE*, WESCON, San Francisco CA, 1970

[3] Boehm B.W., A Spiral Model of Software Development and Enhancement, IEEE Computer, 21, 5, May 1988

[4] Boehm B.W., Software Engineering Economics, Englewood Cliffs NJ, Prentice-Hall, 1981

[5] Brooks F.P., No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer 20, 4, 1987

[6] Mahmood Z., Core Requirements for a Degree Course in Software Engineering, in: *Proceedings of Software Engineering in Higher Education*, Alicante, Spain, Nov 1995.

Curriculum Vitae

Name: Dr Zaigham Mahmood

Email Address: Z.Mahmood@derby.ac.uk

Position/ Institution: Senior Lecturer - Computing
University Of Derby, UK

Qualifications: BSc, MSc, MSc, Ph D, CEng, MBCS

Publications: Core Requirements for a Degree Course in Software Engineering, *Proc. SEHE'95*, Alicante, Spain, November 1995.

Issues in the Teaching of Software Engineering, *Proc. ISSEU'97*, Rovaniemi, Finland, March 1997.

A Functional Approach to Procedural Paradigm for Teaching Software Engineering, *Proc. INSPIRE'97*, Gottenburg, Sweden, August 1997.

Software Engineering in the UK – A Brief Report, *Proc. EuroSPI'99*, Pori, Finland, October 1999.

Experience: Industrial - 17 years
Educational (Lectureship) - 10 years

An Overview of the SPI Activities in Estonia

Ahto Kalja[1] and Jaan Oruaas[2]

[1]Institute of Cybernetics
at Tallinn Technical University
Akadeemia 21, 12618, Estonia
ahto@cs.ioc.ee

[2]Estonian Information Technology Society
Kiriku 6, 10130, Estonia
jaan@eits.ee

Introduction

Estonia among many other nations has seen information technology (IT) as an important tool to improve the case of extremely fast recovery of Estonian economy. This paper gives an overview of the development of IT and the SPI activities in Estonia. After short introduction into Estonian IT history, the IT industry situation is presented. The state of the art of the SPI activities is given.

History of IT in Estonia

The first computers in Estonia were manufactured and installed at the end of 1950s and in the beginning of 1960s. The first computer centres were established in University of Tartu (1959), Institute of Cybernetics (1960) and Tallinn Technical University (Tallinn Polytechnical Institute). The computers installed in these centres were the first- and second-generation Soviet-made Ural and Minsk computers that were used both in scientific research and IT education at universities [1]. The teaching of programmers and other computer specialists started very soon after the computer centres were opened. In addition to popular Fortran, Algol and assembler languages, Estonian researcher groups developed different unique programming languages – Malgol, Velgol etc. These languages had Algol-based structure and were developed for special purposes as for teaching programming languages, solving financial problems etc. In the 80s specialists from Estonia participated in the development of standard software engineering, CASE tools, etc for different ministries of the Soviet Union [2].

Estonian IT Industry

The independent Estonia started eight years ago in a situation, where every ministry had its own Computer Centre developing the information infrastructure of the branch. The first private companies started as soon as it was permitted - at late 80's. At the beginning there were just small co-operatives. Most of the information processing projects used Soviet-made IBM 360/370 compatible computers and the corresponding software. Then the situation changed. At the beginning of Estonian independence (in 1991-1993) a lot of large IT projects in public administration sector were started and a number of private software and hardware companies were established. It was the time of intensive invasion of IBM PC clones and FoxBase(Pro) (pirate-copies as a rule) DBMS into all sizes of software projects. During the next period (1993-1996) a lot of new software and technical facilities were purchased and updated that brought about a rise in the quality of information technology (IT). It was the period when the fight against pirate software was started. Development during last years (1997-1999) has created a situation where most average and large projects use local area networks, client-server architecture, support data-warehouse paradigm etc. The percentage of using licensed software is increasing. All these changes are supported by the new legislation.

Estonian Software Companies

There is about 250-300 IT companies in Estonia. We can say with satisfaction that half of them are active in developing original software. The other half of IT firms are dealers of big western companies, PC producers etc.

A fact worth mentioning is that the two largest Estonian software firms are the information technology divisions of the two biggest Estonian banks (Hansabank and Estonian Union Bank). The IT divisions of these two banks employ over 100 programmers. Other software companies are mostly small, comprising 20-50 people. If we want to compare the software development processes at these companies, we should emphasise that the banks have the best available technical facilities. The IT divisions of banks don’t have notable problems with financing. It means that they can purchase the newest computers (Sun Microsystems server Enterprise 10000 (Starfire) in the Union Bank and HP equivalent systems in Hansabank) and the latest versions of such DBMS as Oracle, Sybase etc. It also means that the software development in these divisions can support all the programming novelties such as component technology, three level systems, data warehouse paradigm, Internet-banking etc. Last and not least - the banks employ a lot of Estonian best IT specialists. For example, Tallinn Technical University has lost dozens of very high-educated employees to the banks. Of course the software developing processes at bank divisions have their own problems as well. At one time the technological process seems as endless improving, at other times as a battle with fire and the project management and documentation tasks are not always solved the best way.

A group of companies that work on a good level thanks to the technological support from abroad are the representatives of large Western companies, such as Microsoft, Oracle, IBM etc. In this case the one who takes care of the software development technology and the training of people is the mother company.

Another group of companies is of kind that works to develop software for western clients. The work of these companies depends on the quality. It means that to survive and preserve clients they have to maintain the same quality level as western software houses. Regretfully these companies at the same time reduce their expenses, for example, in the way that they produce the software without project documentation!

Yet another group of companies works only for Estonian market. This group includes firms, which produce, for example, financial software for Estonian companies, Estonian language-specific text editors, develop Estonian registers or databases etc. Some of these companies are working very successfully and their projects’ software development processes maturity level is often “repeatable”. Sometimes their processes (training processes, configuration management processes etc.) reach even the “defined” maturity level.

Unfortunately there are also many companies whose software development activities take place at the “ad hoc” level. The main reason for their survival is the shortage of qualified IT people and companies in Estonia.

The SPI Activities

No software process improvement plans have been established in general for IT companies and for IT departments of big companies (for example for IT divisions of banks) in Estonia. All the special improvement activities have appeared as ad hoc events. Such types of events include the participation of Estonian specialist in SPI courses abroad, western specialists’ lecturing at Estonian universities or at IT conferences in Estonia, interest of some companies about special improvement standards etc.

The Software Engineering Institute's Capability Maturity Model (CMM) for software process assessment, improvement and capability determination was the first methodology, which was learned and used to improve software processes in Estonia. The people from TTU’s Institute of Informatics and the Abobase Ltd were the first users of this technology in Estonia.

The next model, SPICE, as described below, is the next methodology, which has been introduced to Estonian specialists. Some leading software companies are striving to preparedness for ISO 9000 certificate. One computer manufacturer has it already for two years.

Tallinn Technical University has good contacts with Pori School of Technology and Economics. The delegation (3 people) from TTU visited on December 9-10, 1998 the Software Process Improvement Center (SPIC). Negotiations showed that there is an interest and possibility to organise a similar center in TTU and to start the co-operation in the field.

A centralised activity co-ordinated by governmental organisation is translating IT field ISO standards into Estonian language. Same standards have been accepted on title page methods, but the largely used standards, for example ISO/IEC 12207, Information technology – Software life cycle processes, have been fully translated and versions in Estonian accepted.

SPICE Courses

The idea to introduce SPI international experiences to Estonian IT specialists and companies arose at the beginning of 1998 from Finland. On one hand, the Software Engineering Centre OY and Software Technology Transfer Finland OY with Estonian company Software Engineering Centre AS decided to organise special SPICE standard courses for Estonia, but they were not able to find the minimum necessary number of participants. On the other hand, the Pori School of Technology and Economics is supervising a special Finnish-Estonian Bilateral development project “Teaching of Information Technology on the Bachelor, Master and Doctoral Level”. The aim of this project is to transfer the Finnish experience to improve the role of University research and education from the point of view of the surrounding society; with the focus on software engineering but also some related areas were covered [3].

By joining the resources of these two projects it became was possible to organise the first special course (seminar) on Software Process Improvement and Capability dEtermination (SPICE) = ISO15504 in Estonia. The lecturer was Risto Nevalainen, from STTF Oy, Helsinki. This event was organised in Tallinn on April 2nd-3rd, 1998. Seven participants registered from Tallinn Technical University, 1 from University of Tartu, and 6 specialists from different companies (Aetec Ltd., Abobase Systems Ltd., etc.).

The seminar included the following topics:

- Improvement of software process using different approaches and models (ISO9001, CMM, Quality Awards, SPICE) overview and comparison of the models.
- Use and application of SPICE in software process assessment and improvement planning. Lectures and practical training using real-life case.
- Examples and research results of process improvement activities. Real examples from Finnish companies SPI plans.
- Different approaches and routemaps in SPI for small companies.
- Support in using SPICE standard in small companies.

This first seminar informed several IT specialists in Estonia on the possibilities and needs for software process improvement. The next similar seminar on December 17-18, 1998 was already much easier to organise. Again STTF Oy, Finland was in charge with Estonian Information Technology Society, Institute of Cybernetics at TTU and SEC Ltd, Estonia acting as co-organisers. The seminar gathered 20 participants: 10 from Tallinn Technical University and the rest (developers, experts, managers) from different software companies. Many of them work today in Estonia as process and quality developers and quality consultants. The following companies were represented: Proekspert Ltd, Aprote Ltd, Datel Ltd, Aetec Finantsvara Ltd. As the result of these two seminars Proekspert Ltd, who is DBMS Sybase representative in Estonia, decided to start to use the SPICE standard for software quality improvement. Proekspert also participated in the next SPI project - the INSPIRE.

The INSPIRE Project

The Estonian Information Technology Society participates in the project: The Initiative for Software Process Improvement – Regions Exterieures (INSPIRE). The INSPIRE project is supported by the European Commission as part of COPERNICUS program within the European Software and Systems Initiative (ESSI). The project was started on 1997 and has reached now its final stage.

The objectives of the Initiative for Software Process Improvement - Regions Exterieures (INSPIRE) project are focused upon providing access to the experience and knowledge of various software process assessment, improvement and certification methods, currently available in Western Europe, to SME's[2] from the Central and Eastern European regions.

This focusing is caused from rapid political changes in these regions where the economies are now being committed to migrate to become market driven. As a result, companies from all industrial sectors must modernise their operational practises in order to be successful in competing for new tasks.

INSPIRE recognises that information technology, and software, in particular, will be one of the critical factors impacting upon an organisation's ability to modernise. INSPIRE will target those organisations in which the development of software is of key importance to the success of the organisation.

To address this situation INSPIRE will meet the following objectives:

- to increase the current level of awareness in the benefits of the ISO 9000 Quality Standard and Software Process Improvement (SPI);
- to provide a structured series of SPI training events to facilitate the improvement process;
- to perform a focused series of Process Improvement Experiments (PIE) promoting the use of SPI and Software Process Assessment (SPA) techniques;
- to encourage co-operation between the participating organisations to share both knowledge and experiences;
- to create a basis for permanent links between these organisations leading to continued mutual benefit;
- to actively participate as a member of the Training Cluster, co-ordinated through ESSI project No. 24035: RAPID, disseminating information about the lessons learnt throughout the project's duration.

Consequently INSPIRE will comprise much more than ordinary dissemination - it will perform SPI Training and PIE's.

INSPIRE operates in Estonia, Hungary, Poland and Romania.

Four companies and mentors for some of them have taken part in the project. The companies were targeted to find out their critical business or development processes and make improvement plans. Two of them were software companies, one was a system integrator with its' own development department and the other a system integrator oriented on services and training. The following processes were improved:

- software development process for tracking and description in internal standards within the company with the aim to optimise the process and make it clear for every participant;
- software development and client services assessment using SPICE method;
- business process assessment for ISO9000 preparedness.

[2] Small and Middle-sized Enterprises

All the companies found the results of PIE very successful and the used methods' useful even in cases of very small companies. The PIE made the business processes (i.e. future of the company) clearer for the managers.

The results of PIE have impact on companies from various sides. The experiment showing that typical business software development and supporting process could be easily investigated and assessed could be considered as technical impact. The fact that the Estonian companies’ software capability level profile reached the ISO15504 level 1 and 2 is even more important.

Measuring the following goals at the end of the project: extensions of business activities with clients, meaningful reduction in the average production time of the software in the project etc. have impact on the companies’ business side.

From the organisational side no large changes were made for the experiments. The companies’ employees did some overtime and participated in a few special courses.

The INSPIRE project had also a cultural impact. It showed in people accepting positively the concepts, technologies and management changes concerning software development and more generally in the company production improvement. Anyway all the software teams, mostly composed from young employees with spirit open to innovations, accepted the challenge and moved along in introducing changes to the old software practices.

Last and not least the impact on skills should be mentioned. During the project members of software teams gained significant and valuable new skills like how to use software project management tools according to established procedures. Many engineers involved in the engineering processes received extra training in the software project management, in using assessment standards etc.

Future Plans

Our activities in the field have lead to a deeper understanding of the SPI methods. At the same time several successor projects have been planned.

1. Estonian Information Technology Society is intends to join to the new Software Process Improving plans of EU;
2. Tallinn Technical University is prepared to co-operate more largely with the Software Process Improvement Centre in Pori;
3. Estonian IT and special software companies are planning to continue software process assessment and improvement with the help of CMM and SPICE;
4. Estonian IT standardisation committee has undertaken a task to adapt different international software quality standards.

Conclusion

Developing and running the quality system has helped our IT companies to control the development of our software and quality process. For the future a set of several successor projects have been planned. Inspired by the success obtained during first experiments, Estonian companies intend to continue its software improvement policy in the framework of more general quality strategies.

References

[1] H. Jaakkola, A. Kalja. Estonian Information Technology Policy in Government, Industry and Research. In: Technology Management: Strategies and Applications, Vol. 3, No. 3, 1997, pp 299 - 307.

[2] A. Kalja, J. Pruuden, B. Tamm, E. Tyugu. Two Families of Knowledge Based CAD Environments. In: Software for Manufacturing, North-Holland, 1989, pp 125 - 134.

[3] H. Jaakkola, A. Kalja. Improving IT Education in Estonian Universities. In: Portland International Conference on Management of Engineering and Technology, PICMET'97, Oregon, USA, July 27-31, 1997, pp 276 - 279.

Session 9
SPI and Establishment of Processes/Models II

Chairman

Yingxu Wang

IVF, Gothenburg, Sweden

Configuration Management Deployment & Practice Experiment

Silvia Mazzini
Intecs Sistemi, Pisa

Introduction

The CMEXP (Configuration Management Deployment & Practice Experiment) project is a process improvement experiment (PIE) funded by the European Community (ESSI Project 27637), conducted by Intecs Sistemi.

The overall objective of the experiment is to improve the practice of Configuration Management, by the adoption of state of art Configuration Management practices, based on a sound and ready available technology (ClearCase).

From the point of view of achieving specific measurable objectives, the target is to enhance the Intecs SPICE CM practices profile, from the current Level 1 (performed practices, informal process) "Partially Performed" mark to a "Fully Performed" Level 2 (managed process) mark and at least a Level 3 (defined and standardized process) "Partially Performed" mark.

We expect that the establishment of mature CM practices bring, per se, important benefits to the effectiveness and predictability of the overall Intecs Sistemi Software Life Cycle Process and to the quality and maintainability of its products. Furthermore, the experience gained from the PIE is expected to enhance the company know how and image, in terms of competence and user satisfaction, so far contributing to strengthen the Intecs Sistemi offer of Software Engineering consultancy and services.

Starting scenario

Intecs Sistemi have defined advanced software process and product models for Configuration Management (CM) in the context of a number of R&D and Industrial studies. However, these were heavily relying on highly evolved Software Engineering repository facilities provided by the Object Management System (OMS) of PCTE , an established international standard [1] that has so far failed to translate into adequately supported commercial implementations.

On the other hand, though Intecs Sistemi have achieved an overall Software Process maturity rating BOOTSTRAP Level 3.1 (defined and enforced practices) confirmed by a SPICE Assessment [5], actual deployment of Configuration Management (CM) practices scores far below that mark for projects not strictly regulated by space or defense standards.

A thorough investigation for alternatives to PCTE has identified ClearCase as a sound enabling technology for advanced Configuration Management Practices. Besides being a world class selling tool, ClearCase features capture most of the basic concepts of PCTE (attributes, links, schemes) without loosing full compatibility with Unix and binary Unix tools.

The experiment is an important step from plain Unix directories to more advanced SE Repositories oriented solutions. Integration and consistency of CM practices within Intecs Software process and product model are a primary concern.

Intecs Sistemi staff with the involvement, as consultants, of experienced SPICE (ISO IEC 15504) assessors (QUALITAL) mostly does the experiment. The Intecs Sistemi persons involved in the PIE are:

- one Project Manager
- one CM Administrator
- two Process Engineers
- three Software Engineers

All persons above are involved part-time over the project time span; the Software Engineers are qualified computer specialists, members of the baseline project.

Company Context

Intecs main area of business is the development of system applications and the provision of consultancy services to other organizations operating in advanced technical domains such as aerospace and telecommunications. Another Intecs business sector is the development and marketing of CASE software products. The PIE has particular relevance on this business field, even if a significant degree of transferability of the experience to other business areas is expected.

Configuration and Version management are the keystone of Application Management (AM) for software artifacts that have to exist in the different temporal versions and for different environment configurations; AM has to keep track of many types of product components (code files, documentation, test procedures, SPRs, etc.) and of complex relationships (dependency, composition traceability etc.) among them. A further goal is to provide support for distributed development: Intecs Sistemi has premises in Pisa, Napoli, Roma, Piombino and Toulouse (F) and software projects may involve co-operation among teams from different sites.

Intecs have obtained ISO 9001 certification, and their software process maturity has been assessed according to the SPICE (ISO IEC 15504) and BOOTSTRAP models. These process assessment exercises have pointed out Intecs current CM practices as the weakest aspect of the Application Management process. From a qualitative point of view, an apparently disjoined overall picture has emerged:

- various aspects of the AM process have been formalised as enactable models and specific experimental capabilities for Software Problem Report and Change Management have been implemented in the context of an ESSI PIE project (10189 IPTPM) which has shown the viability of using automatic enactment techniques on well defined and confined process fragments;

- Actual CM practices mostly rely on manual procedures requiring a deep knowledge of the applications structure to achieve the necessary process reliability and effectiveness.

- The use of traditional CM tools (like SCCS and RCS) does not provide a satisfactory support mainly because of the lack of support to complex product models as those promised by advanced SE repositories such as PCTE.

- PCTE that was adopted by Intecs as reference technology for CASE repository has failed to establish itself as a standard for adequately supported commercial products. This acted as a principal barrier to the actual deployment of CM, even though it has represented a very valuable conceptualisation and experimentation. playground

From the quantitative point of view, and because of the above reasons, no reliable assessment of CM practices has been finalized. A "Level 1" score for CM practices has been estimated informally, without a proper CM "profile" identification (SPICE Sup-2 process).

The CMEXP PIE aims to combine Intecs experience and state-of-art technology and techniques to overcome most of the problems and deficiencies of current CM practices. In particular, we expect to achieve a fully operational demonstrator of advanced CM facility suitable to be large scale deployed as standard support for Intecs software development activities, as well as fully defined and validated CM process, procedures and guidelines, ready to be integrated into the company ISO 9000 conforming Quality System.

Baseline project context

The baseline project is the UmlNICE internal product development project.

UmlNICE is an integrated toolset providing support for the Unified Modeling Language (UML) [6]. It is based on state of art technology (CORBA, Java, UML) and is planned to be available in various product configurations on a number of platforms - potentially on all platforms providing support for Java and CORBA. UmlNICE is composed of a large number of components, tightly integrated among them. It can be regarded as an open framework for CASE. In addition to traditional requirements for corrective maintenance, UmlNICE is designed to support continuo product evolution for extensions, tailoring, and tracking of market demand, evolution of the method and

enabling technology.
The UmlNICE project follows an "iterative and incremental" development process inspired at the Unified Software Development Process, the process that has been defined to become the standard process to use UML by the authors of [5]. The project is presently in its 9th intermediate, though self-contained, iteration, which is expected to complete by June 1999 and deliver the first commercial release.
The UmlNICE project was started at the end of 1996 and is expected to complete in the first half of year 2000; the project currently involves a team of ten persons and is planned to remain constant up to the end.

Experiment description

The CMEXP PIE aims at improving the contents, extent and maturity of Intecs Configuration Management practices.
The technology introduced by the experiment are the ClearCase configuration management system [8] and the ClearDDTS change request management system [9].

ClearCase is a world class selling tool, giving support for version control, workspace management, build management and process control. Integrated with ClearDDTS for the change control management, it provides a wide configuration management solution.
The ClearCase approach to the multi-version file extends the Unix and the Windows NT file system to make it a real multi-version project repository, in a transparent way. It makes it easy for an organization to deploy ClearCase, without forcing changes in the existing environment, tools, or the way of work.
Based on an intuitive web-based interface, ClearDDTS allows to track and manage both defect records and enhancement requests, can be integrated with the configuration management system and is flexible enough to be adapted to different organization needs. Particular attention has been paid to the modeling of CM processes, products, and roles and to the definition of procedure and guidelines to ensure that their implementation can take advantage by the advanced features of ClearCase.

Beside the configuration management support process, the main processes effected have been the software development, integration and testing and the whole system integration testing and maintenance, with a slight change of the configuration management, project management and developer roles.

Phases of the experiment

Beside the more obviously identified work items related to Project management, co-operation and dissemination, the core CMEXP activities are organized in two phases, the experiment preparation and execution.
The project has started in June 1998, for the duration of 18 months. At present it is in the execution phase.
In the experiment preparation the ClearCase Configuration Management System and the ClearDDTS Defect Tracking System have been acquired and installed at Intecs and an advanced course has taken place at Intecs premises in Pisa. The purpose of the course was to train the CMEXP project team, including people from the Baseline

Project UmlNICE, for the use and administration of selected technologies.

In the meanwhile the current Intecs CM Practices have been assessed with the involvement SPICE assessor consultants, by setting the Initial CM Profile as a baseline reference for the measurement of the CM process improvement.

The key findings of the initial SPICE assessment have confirmed that the CM process was not completely performed at Intecs, therefore an improvement action was needed first in the activities concerning base practices, such as:

i) the development of a general configuration management strategy, implementation and verification criteria,

ii) the establishing of a configuration management system and

iii) the recording and reporting of the status of configuration items.

As far as the capability dimension is concerned, the low rating at Level 1 was a direct consequence of the incomplete implementation of the process, while the general situation at Level 2 was slightly better than expected, due to the existence of documents modeling the various activities. Everything found at Level 3 was a direct consequence of the implementation of the ISO-9001 certified quality system at the company level.

Taking as input the results of the assessment and the current Intecs CM practices, a generic model of the CM procedures and product entities has been defined, to drive and support the correct execution of CM activities.

The activities for the experiment execution have started with the analysis of the Baseline Project UmlNICE and the tailoring of the Configuration Management System and the Generic Definitions to the baseline project specificity, by instantiating specific CM process products, procedures and guidelines.

Being composed of a large number of components tightly integrated among them, the UmlNICE project follows an iterative and incremental development process, where sometime different development activities run in parallel and the software components evolve through different versions, integrated at different project milestones.

ClearCase allows maintaining a unique repository where to collect all the software versions and releases and developing parallel versions that can be easily integrated by merge facilities; on the other hand the CM procedures become more complicated and are more demanding for the configuration manager.

As a preliminary for the exercising of CM practices within the baseline project, the specific CM guidelines and procedures have been tested by carrying out typical development and maintenance activities, in parallel to the main baseline project activities.

After the period of parallel running, the database has been reloaded with the latest UmlNICE code and the baseline project team has started the CM practices. The PIE team is currently providing guidance and support to the baseline project team and monitoring the project activities.

At the completion of the experiment feedback and lesson learned will be collected and processed and a final SPICE assessment will quantify the process improvements achieved.

Two internal dissemination events have been organized at Intecs, with the participation of the PIE and the Baseline UmlNICE projects and other internal software practitioners and managers:

- a presentation of the CMEXP project experiment, as a preliminary of the ClearCase and ClearDDTS course;

- a workshop to present the defined reference (generic) CM Process and Product Models in depth, and discuss CM Procedures with ClearCase and ClearDDTS.

Consultancy during the experiment

Consultants from the QUALITAL Consortium conduct CMEXP initial and final SPICE assessments.

The QUALITAL Consortium is a non profit institution based in Pisa particularly active in the domain of quality and Quality Systems Certification that participate in the SPICE initiative.

The training of the CMEXP project team for the use of the selected CMS and the training of one CMS Administrator for the installation of the software configuration and for related user support activities has been taught "in house" at Intecs by an ARTIS consultant; ARTIS were the ClearCase distributors for Italy.

Resulting scenario

Technical impact

Since the initial period, the adoption of the ClearCase configuration management system and of the tailored configuration management strategy for the baseline project has resulted in the following short-term improvements:

- resource optimization, by maintaining a centralized source database, with a reduction of source copies;
- easier set-up of the development environments for separate developments;
- better support for separate testing and easy integration of separate developments;
- shorter release build time;
- easier build of previous releases.

A side effect has been also the increment of the internal problem reports, both enforced by the process and encouraged by the ClearDDTS web based interface.

Quantitative measurement of the achievement of the PIE objectives encompasses comparing the results of the initial SPICE assessment of CM practices with a final assessment planned to take place at the end of the project. The SPICE assessment results in detailed profiles of practices and allows capturing improvement in contents, extents and maturity level of CM practice.

Business impact

The business results of the application of a more rigorous CM process and the use of an advanced CM system in the development process are already visible as a significant improvement of the quality and efficiency of the development and maintenance processes.

The extent of the improvements, including the quality of the product and the predictability of the maintenance process, can not be quantified yet. They will become more evident when the developed product will be commercially delivered.

Organization impact

The adoption of the ClearCase configuration management system and the defined CM practices have forced a change in the role of the configuration manager, requiring him a deeper involvement in the project. The configuration manager becomes a sort of assistant to the project leader, in a wider range of activities, not only for the configuration management support process, such as control of the releases and of the changes, but also for the set-up and maintenance of the ClearCase environment for the development, integration, and testing processes.
We expect to transfer the PIE experience to other company development projects in the same or other application domains, therefore resulting in an overall enhancement of the Intecs Sistemi software process.
We intend to institutionalize the CM practice as part of the company Quality System.

Culture impact

Up to now the old Unix configuration management system SCCS was used in the baseline project. The development and maintenance are depending on manual and script procedures. Similar CM practices have been in use for many years and are well accepted. The change in the working environment has generated some skepticism and resistance initially.

Skills impact

The staff involved in the project has acquired an improved skill in the field of configuration management in terms of:

- better understanding of CM concepts and practices,
- experience in the installation, configuration and practice of an advanced CM system as ClearCase,
- greater awareness, both at the managerial and technical level, of the benefits of effective CM practices.

Key Lessons learned

Technological point of view

ClearCase is a sophisticated CM system, suitable for large and complex development activities, such as parallel developments, customizations, availability on multiple platforms, bug fixing of previous releases.
The drawbacks are that the costs of purchase, training, installation and administration are high. In particular a project should buy a license for each developer, each license

costing about 4000 ECU's.
Another weak point is that ClearCase is not yet available for some of the new development platforms in use within the company, such as PCs running with Windows or Solaris x86.
To cope with such problems, Intecs Sistemi have investigated other available commercial tools and have short-listed Visual Source Safe and PVCS, as suitable to be used as complementary configuration management systems. Visual Source Safe version 6.0 emerged as the preferred option for small projects based on Microsoft Technologies.

Business point of view

Configuration Management is essential for a process management and control, and an optimized development and maintenance process.
The managers tend to underestimate the CM. Dissemination actions are helpful to make all the company managers and project responsible aware of the add-on and improvements that the introduction of an advanced CM technology and a more formalized process can provide.
Moreover the participation of managers is necessary to reduce the initial negative feeling, easily coming out starting with a new sophisticated CM tool and motivates the team to change the work environment and accept a more formal process.

Strengths and weaknesses of the experiment

CM is typically a complex issue but projects assign to it just a small percentage of the overall budget. The PIE gives the chance to analyze the whole problem and to find proper solutions, applicable also to future projects.
A weakness of the experiment is that the selected CM technology is quite sophisticated, but also expensive. It may result that many projects, having a moderate degree of complexity and a small development team, adopt cheaper solutions, such as Visual Source Safe or PVCS.

Conclusions

We expect to be able to demonstrate the benefits of a more rigorous and formalized CM process and of the use of an advanced CM tool that allows a better representation of product characteristics and the enforcement of the integrity constraints. Having identified the CM process fragment as the weakest point of the overall software development process, we expect a significant improvement of the software life cycle process as a whole, in particular with respect to Application Management activities.
However, CASE product development and maintenance, which is the scope of the baseline project, is among the most demanding class of activities with respect to CM; experiences and achievements can easily be transferred to other fields. In particular, a successful demonstration allows transferring the results of CMEXP to most of Intecs Sistemi software development projects in a number of other application domains.
Achievement of the objectives of the experiment might therefore result in an overall enhancement of reliability and efficiency of the Intecs Sistemi software process, of the

company know how and of its image (competence and user satisfaction). All these aspects are important factors affecting the competitiveness of the company.

References

[1] Baudier G., Gallo F., Minot R., Thomas I., An overview of PCTE and PCTE+, in: *Proceedings of the 3rd ACM Software Engineering Symposium on Practical Software Development Environments*, pp. 107-109, ACM Press, USA, 1989.

[2] Boehm B. W., Software Engineering, IEEE Transactions on Computing, Vol. 2, pp. 1226-1242, 1976.

[3] Boehm B. W., A Spiral Model of Software Development and Enhancement, IEEE Computer, Vol. 21, N.5, pp. 61-72, 1988.

[4] IEEE 1042, IEEE Guide to Software Configuration Management Plans (ANSI), The Institute of Electrical and Electronics Engineers, USA, 1987.

[5] ISO/IEC 15504, SPICE – Software Process Assessment, ISO/IEC Copyright Office, Switzerland, 1998

[6] Rumbaugh J., Jacobson I., Booch G., The Unified Modeling Language Reference Manual, Addison-Wesley, USA, 1999.

[7] Rumbaugh J., Jacobson I., Booch G., The Unified Software Development Process, Addison-Wesley, USA, 1999.

[8] Rational Software Corporation, ClearCase Reference Manual, Version 3.2, USA, 1998 .

[9] Rational Software Corporation, ClearDDTS User's Guide, Version 4.1, USA, 1998

Intecs Sistemi

INTECS Sistemi is a Software-House providing leading-edge technological support to major European organisations in the design and implementation of complex electronic systems. It operates at the forefront of the software market, where innovation, complexity and quality aspects are essential to determine the company success.

During almost 25 years of activities, INTECS Sistemi has achieved extensive experience in the production of software systems as well as software engineering and quality. Such experience has been acquired through a well-established co-operation with most of the major Italian and European electronic industries and the development of proprietary products.

Silvia Mazzini

Silvia Mazzini received her degree in computer science at the Pisa University, Italy, in 1983. From 1983 to 1987 she was with Systems & Management, Pisa, where she was involved both in industrial software development and research projects. In 1987-1988 she was in Delphi, Lucca, , where she was involved in research projects. In 1988 she joined Intecs Sistemi, where she is now senior software consultant.

Her research interests are in the field of software engineering, specifically in advanced software engineering environments, process modeling, and object-oriented technology.

Quality Assurance for NC-Software with the support of Configuration Management

Roland Beeh
Industrielle Steuerungstechnik GmbH, Stuttgart, Germany
Thomas Bürger
FISW Steuerungstechnik GmbH, Stuttgart, Germany

Introduction

This paper points out, that a CM, that plays a key role in the software development process, is essentially important for QA purposes. Within the ESSI-funded Process Improvement Experiment (PIE) InCoMM (Experimental Introduction of a configuration Management Model) this is demonstrated at the example of an open and modular control system, used for machine tools and manufacturing units.

QA depends on CM support particularly in quality inspection activities such as tests. The introduction of a CM system to ISG's software development process was therefore not only important for the coding phase of the software development but also to the quality inspection phase in order to execute efficient quality inspection activities more effectively. Furthermore a well working CM supports QA in quality control and quality planning in several aspects.

The company Industrielle Steuerungstechnik GmbH (ISG) develops and sells software elements as components for open and modular control systems. The customers of ISG are either control vendors or machine tool builders, who use software of ISG (source code and accompanying documentation) for their own controls.

Software Development through functional extensions

The constantly increasing complexity of machine tools and manufacturing units place a continuously growing demand on the functionality of Numerical Controls (NC). The resulting increase of functionality of NCs is mainly realised by functional extensions of the software of a NC. Due to the large scope of the already existing basic functionality of a modern NC software (about 200 person-years of development effort), the set-up of a new NC software version is based on the existing basic functionality of previous versions. Therefore, the development process of NC software can be described as a further incremental development of an existing system (Fig. BeBer1).

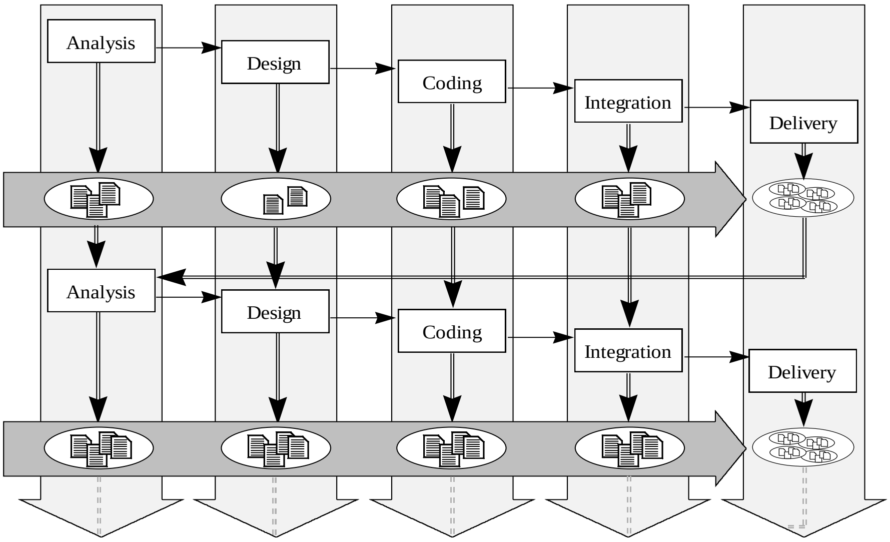


Fig. BeBer1: Further incremental development of NC software

Change activities occur not only due to functional extensions but also to functional improvements and bug removal activities. At ISG those activities are executed parallelly by 22 software developers at one NC software system. In order to enable this further incremental development there exist several customer specific variants (branches) of the NC software system.

Configuration Variety of NC Software

Customers, who put ISG in charge to develop new functionalities within the existing control system, are ready to finance QA activities that relate to this particular functional extension. Additional efforts and costs that may occur with QA activities of the basic functionality of the NC software, have to be paid by the NC software provider. Therefore it is important for the NC software provider to optimise all fields of QA (quality planning, quality inspection as well as quality control) in terms of costs. One of the main problems for testing of NC software are the various possibilities to configure NCs. The resulting large number of configurations illustrates that solely by providing representative test-configurations with an accompanying selection of test cases a high test coverage can be achieved.

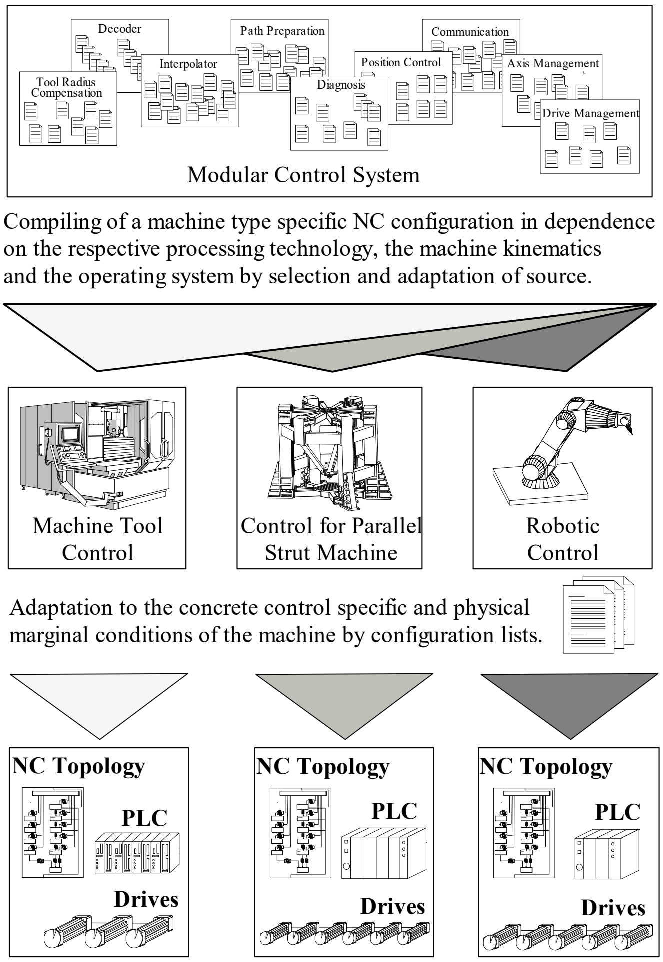

Decoder
Path Preparation
Communication
Interpolator
Position Control
Axis Management
Tool Radius Compensation
Diagnosis
Drive Management
Modular Control System
Compiling of a machine type specific NC configuration in dependence on the respective processing technology, the machine kinematics and the operating system by selection and adaptation of source.
Machine Tool Control
Control for Parallel Strut Machine
Robotic Control
Adaptation to the concrete control specific and physical marginal conditions of the machine by configuration lists.
NC Topology
PLC
Drives
NC Topology
PLC
Drives
NC Topology
PLC
Drives

Fig. BeBer2: Set-up of a modular control system

The sequence of operations to set-up a running NC that is based on a customer specific variant of a modular control system contains two fundamental configuration steps [1] as mentioned in Fig. BeBer2. In the first step, a NC configuration is compiled out of the source code modules, which are available in the respective variant. This configuration is dependent on the respective processing technology (e.g. milling, grinding, etc.), the machine kinematics used as well as the applied system software (operating system). Secondly, the NC is adapted to the concrete control specific (e.g. PLC-interface) and physical marginal conditions of the machine by a dynamic configuration (interpretation of ASCII-lists) [2].

Functional Tests with NC Software

To gain a high quality of NC software ongoing quality inspections are indispensable. Apart from tests with simulation systems [3] and real machines, functional tests, which can be carried out at an earlier stage in the development process within the software development environment, are especially important for the verification of functional completeness [4]. This functional completeness needs to be checked every time a change at the existing NC software was executed. Examinations within the InCoMM PIE at ISG [1] pointed out that about 25 % of all registered bugs can be explained by wrong or incompletely formulated requirements respectively by faults within the specifications. However 75 % of all registered bugs within the NC software arise during coding, integration work or maintenance. Each of this activities, which come up with functional extensions, functional improvements and bug removal activities, imply changes at the existing NC software. Regression tests are particularly suitable to discover bugs, which result from changes within functionality of already existing NC software.

Test Automation for Functional Tests

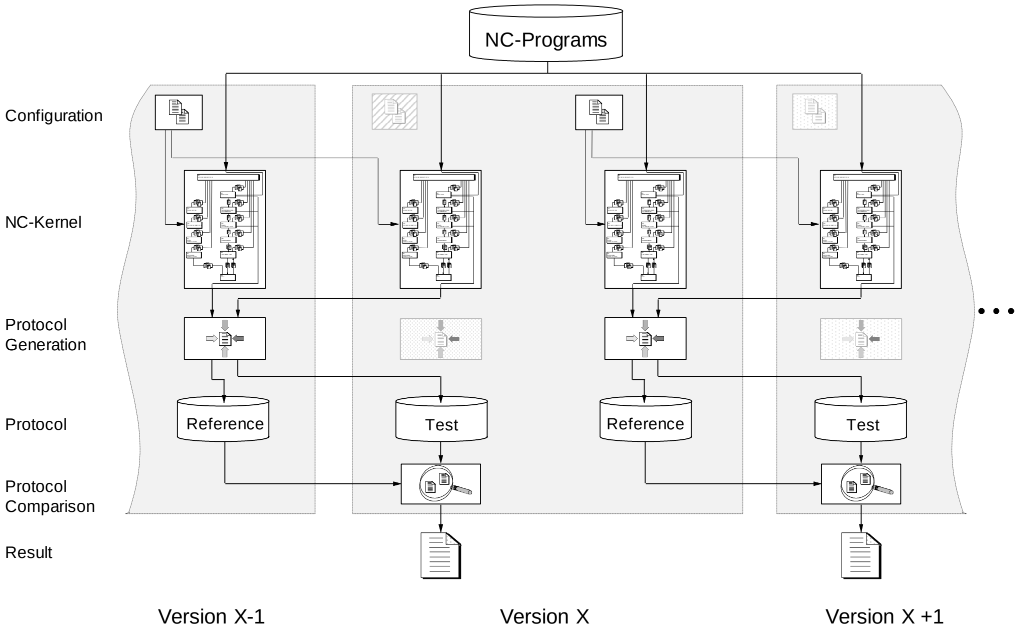


Fig. BeBer3: Sequence of an automatic functional test

At ISG the described sequence of a functional test is carried out by an automated process (Fig. BeBer3). First the test cases of the respective test run and the respective test configuration are selected from a test data base and compiled to a test script. The test cases are formulated as NC programs (e.g. according to DIN 66025) in the same way they are used in the production with NC machines. In the subsequent test run those test cases (NC programs) are sequentially processed by the NC. At the same time the NC also generates function block protocols and test run protocols (logfiles). Function blocks describe the internal data format (process data and control data) of a NC and represent the test output of the respective test case within a functional test.

To be able to compare the function block protocols of the test version with reference protocols of an already tested version, the test version is configured and parameterised identically to the reference version. Because of that and due to the use of a standardised protocol routine it can be guaranteed that all differences, which are identified during the comparison of the actual function block protocols with the reference protocols, are due to changes within the functionality of the NC software. Those identified differences can be explained either by planned changes in the source code or by bugs.

Test Integration in the Software Development Process

At ISG tests are usually executed within the development process of NC software by the individual software developer while implementing a new functionality for the verification of the new functionality. At the end of the development of a new functionality the new functionality is tested with the aid of simulation systems and real machine tools for validation purposes. Furthermore the above described functional test represents the main area of quality inspection activities. It is used for the verification of the existing basic functionality of the NC system after a change (functional extension, functional improvements or bug removal activity) within this system.

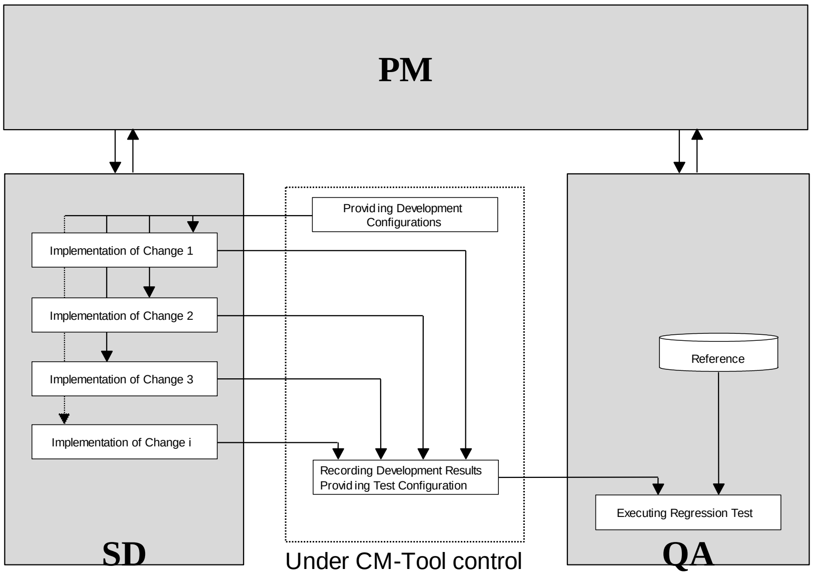

PM
Providing Development Configurations
Implementation of Change 1
Implementation of Change 2
Implementation of Change 3
Implementation of Change i
Reference
Recording Development Results Providing Test Configuration
Executing Regression Test
SD
Under CM-Tool control
QA

Fig. BeBer4: Initial Situation at the Beginning of the InCoMM-PIE

At the beginning of the InCoMM PIE this efficient way of testing was established in ISG’s software development process in an ineffective way. As the administration of the NC software did not support the access to exactly defined NC configurations the functional test was restricted to configurations that could not be assigned to individual implementations (coding and integration) of changes. Several implementations were centrally recorded and were together provided to the QA. The results of a regression test run could therefore hardly be allocated to individual implementations and were badly to interpret due to the large number of differences between actual function block protocols and reference protocols.

Improvement through introduction of a CM System

During an examination of ISG’s software development process possibilities for an optimisation have been recognised.

In the field of QA it became obvious, that more administrative support is required in order to execute the existing functional tests in a more effective way. Beside of this missing support for quality inspection additional possibilities for an optimisation of quality control and quality planning could be formulated.

As the number of customer specific NC variants and therefore the number of parallel developments at one common NC software system are increasing the field of software development (SD) at ISG needed also additional assistance in administering software elements of the NC system. Improving this assistance an increase of software elements' reusability, an avoidance of software elements' redundancy and faster responses on support requests from customers shall be achieved.

So the examination pointed out, that an administrative instance for the support of SD and QA as well as for the interaction between SD and QA is missing at ISG. Therefore ISG intends to improve the software development process within the ESSI-funded PIE InCoMM by introducing a CM system to ISG's software development process, which is based on the guidelines of the V-Model [5].

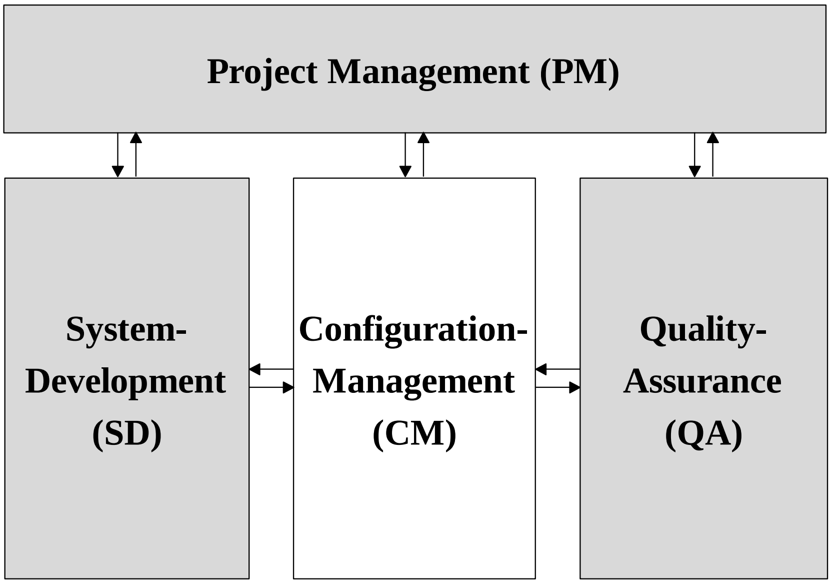


Fig. BeBer5: CM as Interface in the Software Development Process [5]

In accordance with the V-Model CM is considered as an independent field of activity in the software development process, which is equal to the fields SD, QA and project management (PM) (Fig. BeBer5). Concerning the responsibility CM can be considered as centrally placed interface in the software development process.

For the introduction of the CM system ISG follows the CM definition in accordance with [6] in order to fulfil the mentioned requirements of SD and QA:

The field of activity of configuration management includes technical and organisational based procedures, with which

- *the identity of specified software elements at certain moments in time is determined and*
- *the modification of those software elements is a controlled and reconstructable procedure.*

In this context, the term software element describes either the smallest definitely identifiable part (e.g. a source-module) of a software configuration, or a software-configuration (e.g. a software product) which is assembled by several software elements [7].

In summary ISG expects the following improvements by the introduction of a CM system as central administrative instance into the software development process:

- Improvement of effectiveness of functional tests
- Earlier detection of bugs
- Simplification of bug tracking
- Reduction of effort for bug removal activities
- In-time delivery of software
- Higher customer satisfaction

Effective QA with the Support of CM

Tool application for CM within the NC Software Development

At the beginning of the InCoMM-PIE ISG considered to introduce tools out of the following tool categories:

- Version oriented tools (e.g. PVCS, MS-SourceSafe),
- Developer oriented tools (e.g. ClearCase, MKS Source Integrity) and
- Process oriented tools (e.g. Continuus, CCC/Harvest, PVCS Process Manager).

The emphasis of the version oriented tools is on the pure version control of software elements (uniquely identification of successive versions). In addition to the features provided by the version oriented tools developer oriented tools support software development in teams. The definition of so-called views allows to arrange configurations existing out of software elements with different versions. Furthermore these tools support good code-merge capabilities as it is frequently necessary for parallel software development purpose.

As the ISG has not yet defined formal and restrictive processes for software development, it turned out that process oriented tools are unsuitable for ISG at the current situation. ISG’s main demands on a tool are the support of parallel development and the supply of machine type specific NC configurations. With the assignment of attributes views for individual software developers or whole developer teams can be defined. Thus, both individuals and groups are enabled to use the same software elements, which exist in different variants. Due to this facts ISG decided to evaluate the developer oriented tool ClearCase from Rational for the period of the InCoMM PIE.

The technical introduction of ClearCase to ISG’s development environment (MS-Visual C/C++, Win 95, WinNT) involved no remarkable problems. The complexity of ClearCase requires one main responsible for the tool whose effort for administration needs to be taken into account and to be planned. This complexity is also reflected in the great number of possibilities to automate sequences of events by so called triggers. To implement these possibilities Perl scripts have to be written with considerable effort.

Within the scope of the InCoMM PIE ISG decided to realise a change management database within the already used Intranet Lotus Notes for the support of the change management process. The underlying process corresponds to the change management process defined within the V-Model [5] as mentioned in Fig BeBer6.

The assignment of ClearCase and the change management database to the individual CM activities is described in Fig BeBer6. The interaction between ClearCase and the change management database is not yet realised via software interfaces. This is one of the goals for further improvements of CM at ISG.

Set-up of CM for NC Software Development

In compliance with the approach of the V-Model [5] the first step of the set-up of CM was to define, which CM specific activities and products are relevant for the development of the modular control system of ISG. Subsequent to this so-called Tailoring an investigation took place to find out, which of these activities and products are already realised and existing at ISG. The activities of CM (Fig BeBer6) according to the V-Model are

- CM-Planning,
- Product- and Configuration Administration,
- Change Management and
- CM-Services.

CM-Planning (e.g. the definition of processes and guidelines) and CM-Services (e.g. back-up of results) are activities, which are once generally defined respectively installed for all projects of ISG. However Product- and Configuration Administration as well as Change Management are integral parts of the software development process and are therefore the essential CM tasks.

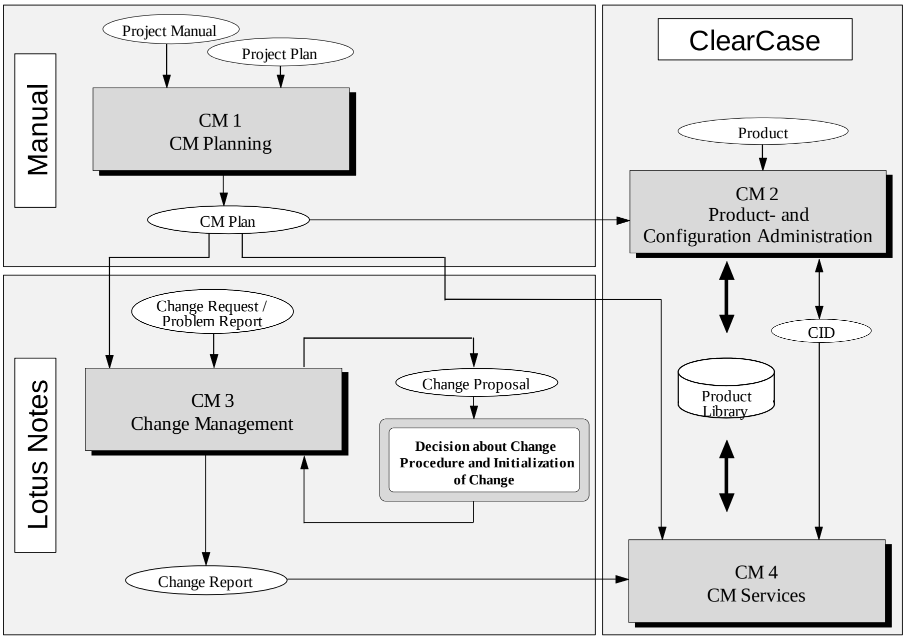


Fig. BeBer6: CM activities according to the V-Model [5]

Establishing the ***product- and configuration administration*** to the software development process improves the customer support. The number of parallel existing customer specific variants of the NC system (branches) can be increased and therefore a clear arrangement of developments guaranteed. Using an appropriate numbering and identification system for versions and variants all customer releases, which are delivered as source modules, can be reproduced in a fast and reliable way in case of support requests by the customers. The compilation of customer specific variants can be accelerated by the use of a suitable CM-tool like ClearCase. ClearCase also supports the bug tracking in an effective way. Due to the possibility to view changes at the line-of-code level [8], the time and responsibility of a known bug can be comfortably evaluated. All this leads to a higher customer satisfaction in the co-operation with ISG.

As already mentioned the development of NC software can be considered as further incremental development of an already existing control system. Therefore further functional extension, functional improvement as well as bug removal activities are always connected to changes of the existing NC software. For this reason, the ***change management*** is especially important for the development of NC software. With using a defined process [5; 7] the demand for controlled and reproducible changes can be sufficed. ISG uses a database, realised within the Intranet Lotus Notes, to support the change management.

Due to the set-up of the ***change management*** to the software development process of ISG the co-ordination of parallel developments can be improved. Consequently the described automatic functional tests for the different implementations can be executed in a more effective way. Problem reports by the customers can be assigned to the individual changes within the NC system as all changes are registered and administered separately by the change management database. The resulting reduction of effort for bug removal activities leads to a faster and more effective customer support. The improved co-ordination of parallel developments results also in the avoidance of unnecessary and expensive development of identical changes in different customer specific variants (branches) by different software developers.

Interaction of QA and CM

In the field of quality inspection the introduction of CM to ISG's software development process leads to an improved use of the automatic functional tests. As mentioned in Fig. BeBer7 the implementations of changes are now considered to be parallelly executed activities. The results of these individual change activities can be separately provided to QA for quality inspections such as functional tests. This is possible due to the administrative support for product- and configuration administration with ClearCase and due to the change management database.

During the coding work for a particular change the originally provided configuration of the NC system changes due to the integration of other changes into the common NC system. To improve the integration of a change CM provides the software developer with an integration configuration (up-date) to integrate (merge) the new code. In order to reduce the overall effort for the execution of regression tests for individual changes ISG intends to subdivide the regression test into two steps. In a first extensive test run the functional correctness of the changes can be examined whereas in the second test run the correct integration of the changes into the current version of the NC system can be proved.

In addition to the support for quality inspection support CM supports QA also in the fields of quality planning and quality control. As with the introduction of the change management data base all changes within the NC system become planned activities also the quality related activities can be planned. Suitable test cases can be selected from the test data base for regression tests or reviewer for code reviews can be included in the plan at a determined time. Basing on the results of the quality inspection activities and on the statistical evaluation of problem reports as part of the change management database quality control can be done. Software parts of the NC system with increasing bug rates can be detected or activities, which are susceptible to mistakes, can be identified. This allows the selective and planned use of appropriate quality inspection activities.

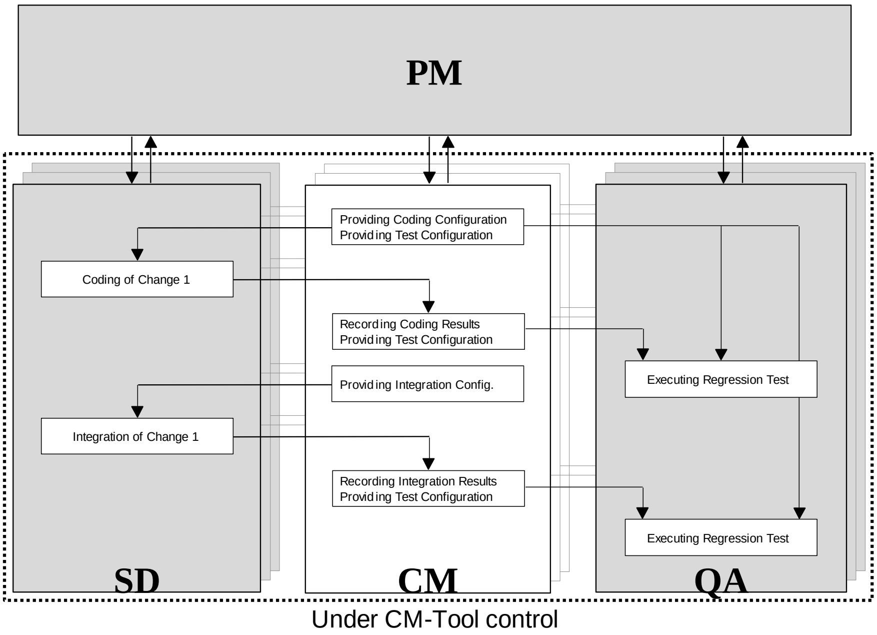

PM
Providing Coding Configuration
Providing Test Configuration
Coding of Change 1
Recording Coding Results
Providing Test Configuration
Providing Integration Config.
Executing Regression Test
Integration of Change 1
Recording Integration Results
Providing Test Configuration
Executing Regression Test
SD
CM
QA
Under CM-Tool control

Fig. BeBer7: Interaction of QA and CM for Quality Inspection

Key Lessons learned and Expected Impacts

The introduction of CM to ISG's software development process with the involved interaction of QA and CM led to a quality improvement of the NC software delivered to customers. This can be proved by the reduction of the percentage of bugs detected by customers from 26 % to 18 % since the beginning of the PIE in July 1998. This trend towards a higher delivered quality is expected to be steady in spite of the continuously increasing number of parallel developments. Among other thing this earlier detection of bugs can be led back to the increasing effectiveness of functional tests due to the introduction of CM.

The main business impact of the InCoMM PIE is the possibility to support more customers faster and in an easier way with NC-software due to the aid of a working CM. Customer specific developments can be co-ordinated, functionality can be merged between the variants, bugs can be traced back in a reliable way and responses on support requests can be accelerated. The number of parallel existing variants (branches) of the NC increased from 27 to 50 since the introduction of ClearCase (April 1999). The expected improvements in customer support (e.g. time necessary for responses on support requests, time necessary for delivery a new NC version) will be

evaluated at the end of the InCoMM PIE (February 2000).
With the introduction of CM to be an independent field of activity the effort for CM related activities became a calculable part in the software development. A tendency towards an overall reduction of CM effort is the expected result of a final cost-benefit analysis at the end of the InCoMM PIE. This will reveal the necessity of CM also from a business point of view.

Conclusion

A successful and effective QA requires a functioning and practicable CM. CM that is assisted by an appropriate tool makes not only the SD easier but supports also all fields of QA (quality planning, quality inspection as well as quality control). This knowledge led to the decision to introduce the CM tool ClearCase for the complete ISG as basic CM tool.
The knowledge, which is gained out of the InCoMM project according to CM in general and CM tools in particular, contributes at ISG in an interlocking system between QA and CM. This will result in an improvement and a higher automation of QA activities. Basing on the use of ClearCase and an optimised change management data base process improvement will be a permanent concern at ISG.

References

[1] Pritschow G., Beeh R., Bürger T., Qualitätssicherung in der CNC-Steuerungsentwicklung, in: *Zeitschrift für wirtschaftlichen Fabrikbetrieb,* pp. 353-356, Carl Hanser Verlag, München, Germany, June 1999

[2] Daniel C., Dynamisches Konfigurieren von Steuerungssoftware für offene Systeme, Springer Verlag, Berlin, Heidelberg, New York, 1996

[3] Meier H., Kreusch K., CNC-Test an virtuellen Werkzeugmaschinen, in: *Zeitschrift für wirtschaftlichen Fabrikbetrieb,* pp. 415-417, Carl Hanser Verlag, München, Germany, September 1998

[4] Liggesmeyer P., Rothfelder M., Ackermann T., Qualitätssicherung Software-basierter technischer Systeme - Problembereiche und Lösungsansätze, in: *Informatikspektrum,* pp. 249-258, Springer Verlag, Berlin, Heidelberg, Germany, October 1998

[5] N.N., Entwicklungsstandard für IT-Systeme des Bundes - Vorgehensmodell, Teil 1: Regelungsteil, Allgemeiner Umdruck Nr. 250/1, Oldenbourg Verlag, München, Wien, 1998

[6] Frühauf K., Unterlagen zum TAE-Lehrgang Software Konfigurationsmanagement, Technische Akademie Esslingen, Ostfildern, Germany, 1998

[7] Scheifele D., Beeh R., Bürger T., Nagel T., Configuration Management for Open Modular Control Systems, in: *Proceedings of the Sixth European Conference on Software Quality,* pp. 526-534, Arbeitsgemeinschaft für Datenverarbeitung (ADV) Handelsgesellschaft mbH, Wien, Austria, April 1999

[8] Tichy W.F., Configuration Management, John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore, 1994

Glossary

CID	Configuration Identification Document
CM	Configuration Management
DIN	Deutsche Industrie Norm
ESSI	European Systems and Software Initiative
InCoMM	Experimental Introduction of a Configuration Management Model for a Open Control System (ESPRIT IV, EP 27546)
ISG	Industrielle Steuerungstechnik GmbH
NC	Numerical Control
PIE	Process Improvement Experiment
PLC	Programmable Logic Controller
PM	Project Management
QA	Quality Assurance
SD	System Development

Appendices

About the Authors

Roland Beeh was born in 1963. He studied electrical engineering at the University of Stuttgart. Since 1991 he is responsibly working in the fields of software development and quality assurance at Industrielle Steuerungstechnik GmbH.

Thomas Bürger was born in 1970. He studied mechanical engineering at the University of Stuttgart. Since 1996 he works as a research assistant at FISW Steuerungstechnik GmbH in the field of machine-oriented control functions.

About ISG

The company Industrielle Steuerungstechnik GmbH (ISG) develops and sells software elements as components for open and modular control systems. The customers of ISG are either control vendors or machine tool builders, who use software of ISG (source code and accompanying documentation) for their own controls. The software of ISG can be extended with their own software modules. This openness results in a great number of software elements such as source code files, documentation, or test patterns.

Being a service provider, ISG places itself at the demands of its customers and employs 22 engineers in the central business field, which is the development and maintenance of NC software. It has an annual turnover of about 4 Mio DM.

ISG was founded in 1987 with the aim to support control vendors and machine tool builders with an open control system in such a way that they can concentrate on their own innovative strength in their fields of competence. ISG's NC software is used in a wide range of technologies. Thus, it is in use for e.g. 3- and 5-axis milling of metal-working and wood-working, for lathes, for grinding, for a various number of technologies with parallel strut machines, for high-speed plotter, robots and robotal devices.

Software Process Improvements in a Very Small Company

Ita Richardson
Department of Computer Science and Information Systems and Small Firms Research Unit,
University of Limerick,
National Technological Park,
Castletroy,
Limerick,
Ireland

Kevin Ryan
College of Informatics and Electronics,
University of Limerick,
National Technological Park,
Castletroy,
Limerick,
Ireland

Introduction

Following any software process assessment, it is important that an organisation implements a process improvement strategy to produce a well-defined software process. In theory, this is simple; it is much more difficult in practice.

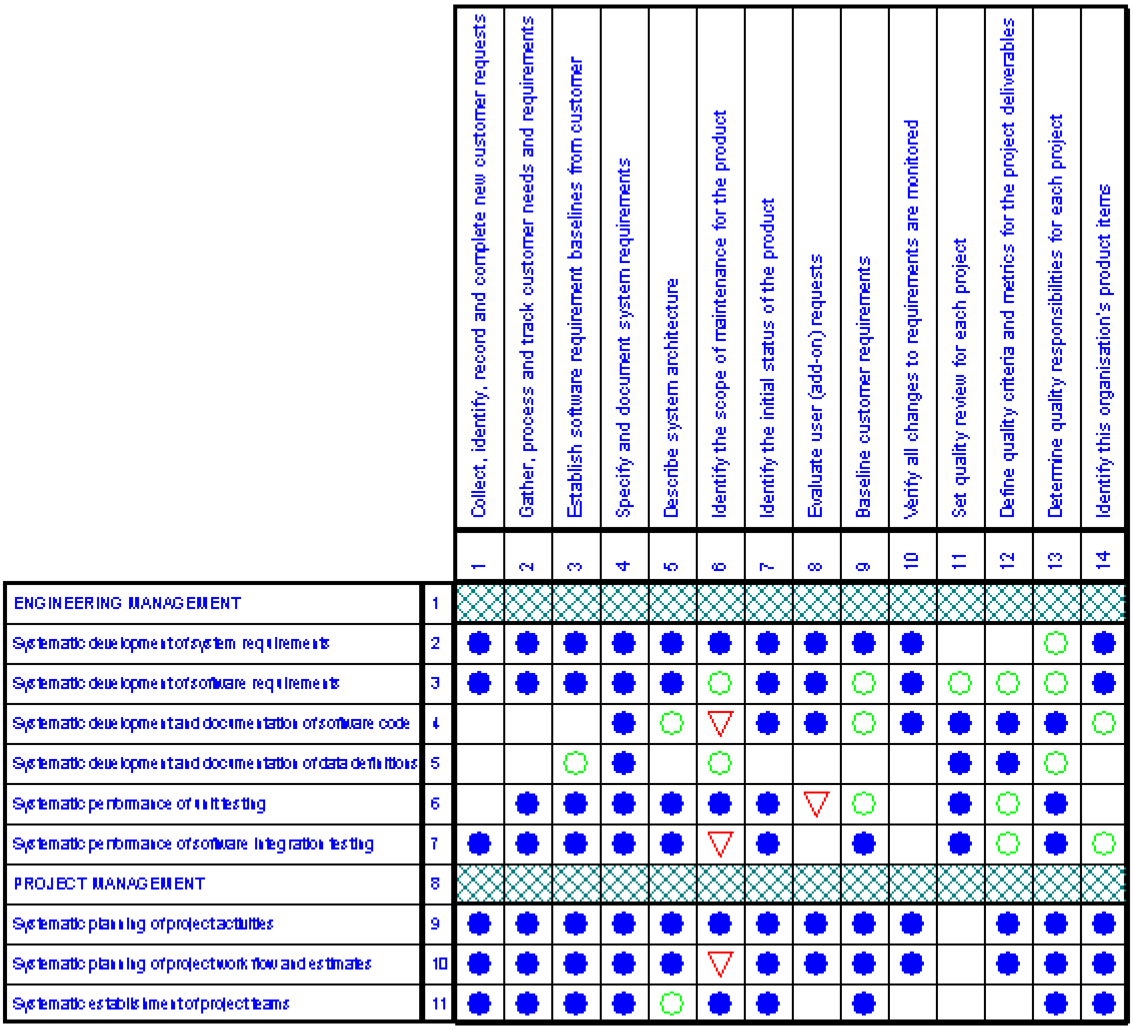
Authors have recognised that the available models did not provide an improvement strategy [1], [2]. The IDEAL model for the Capability Maturity Model was presented in 1995 [3], but as recently as 1998, Debou and Kuntzmann-Combelles note that “due to lack of documentation on the post-assessment phase, assessments are often being performed as a one-shot event without connection to any improvement strategy” [4]. For the large company, it may be possible to devise such a strategy by investing in the development of action plans. However, for the small company, this requires yet another investment from a much smaller purse.

The authors of this paper have developed a generic method to be used by Small Software Development companies, allowing them to devise Software Process Improvement strategies. This method is based on Quality Function Deployment. They implemented the method – Software Process Matrix - in two small software development companies in Limerick, Ireland. The paper discusses this implementation in one of those companies.

What is the Software Process Matrix (SPM)?

The Software Process Matrix is a method used to establish an improvement strategy based on Quality Function Deployment. Strong, medium and weak relationships between practices and processes were identified through hypothesis testing of experts' opinions.

Figure 1



		Collect, identify, record and complete new customer requests	Gather, process and track customer needs and requirements	Establish software requirement baselines from customer	Specify and document system requirements	Describe system architecture	Identify the scope of maintenance for the product	Identify the initial status of the product	Evaluate user (add-on) requests	Baseline customer requirements	Verify all changes to requirements are monitored	Set quality review for each project	Define quality criteria and metrics for the project deliverables	Determine quality responsibilities for each project	Identify this organisation's product items
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
ENGINEERING MANAGEMENT	1	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Systematic development of system requirements	2	●	●	●	●	●	●	●	●	●	●			○	●
Systematic development of software requirements	3	●	●	●	●	●	○	●	●	○	●	○	○	○	●
Systematic development and documentation of software code	4				●	○	▽	●	●	○	●	●	●	●	○
Systematic development and documentation of data definitions	5			○	●		○					●	●	○	
Systematic performance of unit testing	6		●	●	●	●	●	●	▽	○		●	○	●	
Systematic performance of software integration testing	7	●	●	●	●	●	▽	●		●		●	○	●	○
PROJECT MANAGEMENT	8	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Systematic planning of project activities	9	●	●	●	●	●	●	●	●	●	●		●	●	●
Systematic planning of project work flow and estimates	10	●	●	●	●	●	▽	●	●	●	●		●	●	●
Systematic establishment of project teams	11	●	●	●	●	○	●	●		●				●	●

Software Process Matrix

The values given to strong, medium and weak relationships are 9, 3, 1 respectively and on Quality Function Deployment charts they are normally represented by the symbols: l or ◎ (strong), m (medium), and Δ (weak). To use the Software Process Matrix, companies carry out a self-assessment of their software process and input the results. SPM outputs a prioritised action list showing the practices to implement. A sample of the Software Process Matrix is displayed in Figure 1.

Computer Craft Ltd.

Computer Craft Ltd is a software development company based in the mid-west region of Ireland. During the initial stages of the research, the company had two software development groups, one with responsibility for mainframe products, the other for personal computer products (PC development group). The mainframe group was no longer in existence at the completion of the research. Computer Craft Ltd. have a large number of customers, with a larger customer base in the U.K. than in Ireland.

Although Computer Craft started out developing business software itself, in recent years they partnered with a U.S. company, giving Computer Craft the rights to modify and install a software package (Data Organizer) developed by this U.S. company. There are two aspects to the modifications made to this product. Initially, the product was to be localized. For example, Irish and U.K. legislation had to be accounted for. It can also be customized for the needs of the user. If major changes are required by the customer this often becomes a separate module, programmed in Ireland, and retained as a generic feature. This varies from project to project. About 30% of customers require extensive customisation. The Computer Craft engineers install software directly in customer sites.

Research in Computer Craft

This was an action research project in Computer Craft. Employees were interviewed and observed in their work. Access to complete documentation on two projects was obtained. One project was developed in the initial stages of the research; the other was developed towards the latter end of the research. The four processes affected as a direct result of the SPM implementation are discussed in this paper.

Starting Scenario

Organisation Processes

Prior to the researcher visiting the company, there was no particular emphasis on software process improvement. There had been some work done on the development of standards within the software development group, but this was done on a personal level by the manager of the group rather than it being promoted within the organisation. The company employed a software quality assurance engineer whose main focus was on testing.

The fact that both the Group Managing Director and the Software Development Manager were willing to give employee time to the project is significant in itself. However, when recruitment difficulties arose, this project was given lower priority.

Customer Management

Customers were dealt with exclusively by the sales representatives in Ireland and the U.K. It was not normal for developers in the software development group to meet the customer until installation time. The flow of information between the customer and the developer was as shown in Figure 2.

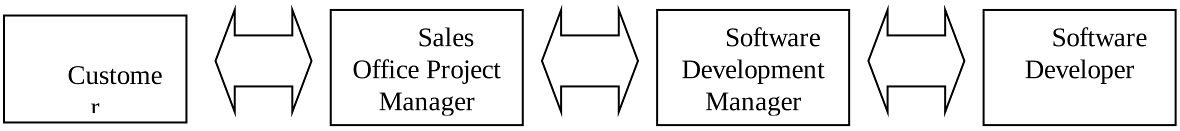


Figure 2
Information flow between customer and developer

For customized product, the customer services project manager wrote up a requirements specification. There was no particular standard in use – *"each customer services project manager identifies the need in their own way*", and complete information was not always received from the client: "*the client may not know how to address particular issues*". Once completed, this document was signed-off by the customer and was passed to the software development group. However, there were cases where "*questions were left unanswered*" and "*information was lacking*".

When the software development group received the requirements specification, a functional specification was written. This specification was reviewed and accepted by the customer. Again, this was done through the customer services project manager. In the opinion of at least one software engineer, they could have "*quantified what needed to be done"* earlier in the project if someone from the development group met the customer. It was not uncommon for work on the functional specification to be stopped until the customer made decisions about their requirements.

Other difficulties arose with the functional specification, where there "*were lots of things missing*", "*it was different to what was tendered*". For example, one problem arose because the person whom the customer services project manager dealt with at the customer site was not the final user of the system. On another project, the software engineer assumed that all the information gathering had been completed for the requirements specification, but this had not been the case.

Customer requirements often changed as developers proceeded through the design stages. One developer stated: "*There is a huge amount of information from the customer, but it is changing – moveable goalposts*". When changes came from customers, there was no formal means of dealing with them. The difficulty faced by the group was that features not previously included in the requirements and subsequently the functional specification impacted the final project schedule.

During some projects, developers met customers at prototype stage, but this was more the exception than the norm, which was that the developer would meet the customer during installation. Not meeting the customer until late in the project caused problems in the development group. Developers found that "*information useful to the design and development was gathered from the customer by the developer during the installation*".

As with requirements specifications, there were neither standards nor templates for functional specifications within the company. One developer told the researcher that he wrote a "*functional specification based on his own experience*". Consequently, each engineer had their own format and layouts and sections within specifications varied considerably. However, when functional specifications were written, the decisiveness of the engineer was usually prevalent throughout them. For example: "*The user will have the option to approve, reject or hold a transaction*", "*after review the user will have a set of transactions that are approved*".

Developers sometimes found problems at later stages in development, such as system test and installation, which could have been avoided if requirements had been collected properly. This was a frustrating situation for the software engineers, particularly as they had deadlines to meet.

Implementation

Decisions about implementation of software were made by the customer services project manager. They needed to examine such issues as: could the product be installed? could it do the critical tasks? were there any show stoppers? or cosmetic bugs? The final decision was usually based on the answer to "*is it a major concern if the software goes with this bug?*" As there was no quality shipment policy defined, "*there have been a few bad blunders*". The quality policy only required that, prior to shipment, all software was signed off by three people and there was a stamp on delivery media.

Following test completion, the software was shipped to the customer services project manager who installed the software for the customer. In some cases the software development manager or software engineer helped with the implementation. When there were changes made during implementation the customer compared the product with the requirements specification, which was unambiguous where they were concerned. Software installation was done either from diskettes or a laptop computer.

Project Management

In Computer Craft, project deadlines were driven by the customer, giving the software development group no leeway or contingency on time. At the start of a project, the software development manager drew up an initial schedule, breaking this into defined phases and milestones, but these were not always adhered to. The main difficulties arose when a schedule was set, and then new features, which disrupted the schedule were added. The manager also drew up a Gannt chart for projects, which were usually kept up to date until the project was approximately 60% complete.

In the early stages of projects, the software development manager decided what were the critical tasks from the initial customer requirements, to whom these tasks were assigned and the time-scales involved. Software developers were often working on multiple projects and were seen by the manager as "*a team working together*". The software development manager was ultimately responsible for the completion of the project.

Group meetings were important within Computer Craft, mainly to keep people informed of progress on development projects. There were two general meetings held each week. One was a production meeting, attended by the two software development group managers (mainframe and PC) and customer services. This was followed by the software development group meeting, at which development tasks were scheduled and updates from the production meeting passed onto the group. Normally, two project meetings were also held each week.

Intervention using Software Process Matrix

Questionnaires on current performance were circulated to the software development group. The software development manager was questioned on current performance, planned future performance and importance to the company. Average current performance was calculated from the results received.

The overall importance of each process was calculated and an example is shown in Tables 1 and 2.

PROCESS	Current Performance	Planned Future Performance	Improvement Factor
Software deliveries and installations	3.2	5.0	1.36
Establishment of project teams	2.8	4.0	1.24
Configuration management	3.4	5.0	1.32

Table 1
Rating Customer Requirements

For the process 'Establishment of project teams', current performance of 2.8 is the average score given by the software engineers on the self-assessment questionnaire. The planned future performance of 4.0 means that the software development manger wants to see these being established by applying an organisational procedure. The improvement factor in Table 1 is calculated as: 1 + (Planned Future Performance – Current Performance) * 0.2, giving, in this case, a value of 1.24.

PROCESS	Improvement Factor	Importance to the Company	Overall Importance	Percentage Importance
Software deliveries and installations	1.36	5.0	6.8	2.5
Establishment of project teams	1.24	4.0	4.96	1.8
Configuration management	1.32	4.0	5.28	1.9

Table 2
Calculating Overall Importance

The importance to the company value of 4.0 means that the software development manager stated that this process was of high importance to the company. As shown in Table 2, the overall importance was calculated by multiplying the importance to the company by the improvement factor. Percentage importance of each process is the overall importance stated as a percentage of the total overall importance in the Software Process Matrix.

According to the results the following were the top nine processes that were important to Computer Craft:

1. Preparation and performance of deliveries/installations
2. Systematic planning of project activities
3. Preparation of the customer for new product release
4. Systematic development and documentation of software code
5. Systematic support of correct and efficient software
6. Systematic management of customer needs throughout life-cycle
7. Focus on markets and customer satisfaction
8. Systematic planning of project work flow and estimates
9. Systematic development and documentation of data definitions

To calculate the importance of each practice, the overall process importance was multiplied by the strength of the relationship between that process and practice, and these were totaled.

PRACTICE	Importance of the Practice	Percent Importance
Transform each software component into software units	175.2	1.0
Assign a person with software quality assurance responsibilities	252.0	1.4
Use evaluation list of approved suppliers	24.4	0.1

Table 3
Importance of the Practices

The importance of the practice was also stated in percentage terms, and those with the highest values were identified as those that would cause the greatest improvement to the software process within Computer Craft. Examples of practices are in Table 3.

The top ten practices proposed by using the Software Process Matrix as the basis for an action plan were:

1. Identify this organisation's product items
2. Establish product baselines for each product supplied
3. Verify all changes to requirements are monitored
4. Specify and document system requirements
5. Collect, identify, record and complete new customer requests
6. Assign a person with SQA responsibilities
7. Identify the initial status of the product
8. Define delivery contents (media, software, documentation, documentation) to customer from subcontractor / software development group
9. Define quality criteria and metrics for the project deliverables
10. Assign responsibility for software development plan, work products and activities.

The software development group in Computer Craft were not willing to accept the actions at face value, and, at a group meeting, chaired by the researcher, there was a discussion as to what should be implemented and how this should be done. All of the practices identified by the researcher using the self-assessment questionnaire and the Software Process Matrix were included in the final action list defined by the company. They felt that some of them should be worked on together, and were not prepared to work on the actions in the priority given without some discussion. At the end of the meeting, the group had decided that the company should concentrate on these practices, combined into action items with the following priority:

<u>Action 1:</u>

- List all the company's products and their dependencies, based on practice 1 above – Identify this organisation's product items. Computer Craft do not have this currently. While it is in people's heads, it is not written down, and members of the group had spent some time that day working on this very issue.

<u>Action 2:</u>

- Set up a procedure to specify and document system requirements, taken directly from practice 4 above. They wanted to have a template for the specification but not a procedure as this would inhibit developers too much.
- Set up a procedure to collect, identify, record and complete new customer requests when new requests come in, prior to development. This was taken directly from practice 5.

<u>Action 3:</u>

- Set up a procedure to define delivery contents (media, software, documentation) for the customer from the subcontractor / development group. This was taken from practice 8.
- Set up a procedure to verify all changes to requirements are monitored once the requirements specification has been signed off (practice 3).

<u>Action 4:</u>

- Set up a procedure to define quality criteria and metrics for the project deliverables – this was the 9th practice listed.
- From practice 6, assign a person with SQA responsibilities and also define what the SQA responsibility is: while individual software engineers must be responsible for their own quality, the SQA person should have a responsibility of letting people know the quality criteria. They needed to examine at the job description, define

what software quality means, think about criteria and metrics, and more than just open, closed, pending bugs.

Action 5:

- Set up a procedure to establish product baselines for each product supplied so that Computer Craft will know what is the minimum product that they will ship from development. This action is based on practice 2.
- Set up a procedure to assign responsibility for the software development plan, work products and activities, based on practice 10.

Action 6:

- Set up a procedure to identify the initial status of the product which could be based on their current handover document. This is practice 7.

Results

This section of the document discusses the processes in early-1999. Much of the discussion centres around the Banking Organiser project, the main project being worked on by Computer Craft in the latter part of 1998, to which updated software processes were applied. This project consisted of a series of software modules which extracted data from Data Organiser, outputting it in formats which interfaced with other software used in the customer company.

Organisation Processes

At the end of the research period, the software development group had reduced in size, and consisted of the software development manager, two software engineers and the quality assurance engineer. The emphasis which the company placed on quality assurance was still evident - when the previous quality assurance engineer had resigned from the company, he was replaced almost immediately, although the size of the software development group had decreased. However, the role of the quality assurance engineer had changed. While she had a responsibility for testing and test plans, she also was responsible for writing up procedures and for the improvement of the software process within the company.

Customer Management

The main project worked on by the software development group was Banking Organizer. In this project, the software development manager dealt directly with the customer. Initially, the customer produced a document listing their requirements. Following this, the software development manager spent some time on the customer site, met their project manager, and attended meetings about their requirements. The Banking Organizer project manager became the point of contact for the software development manager, who then passed requirements to the software developers. If the software development manager could not answer the software developers questions, then he talked to the project manager. As the project progressed, the developers had direct contact with the customer, and recognized that their customer was the company for whom the product was being developed.

Program specifications for this project were written by all members the software development group, all using the available template. The customer project manager examined the documents, and signed off on quotations generated from them. Specifications were passed between developers and all information needed by them was available in the specifications. The document history was updated on some documents the researcher examined, although she was told by the project manager that this was something he did not follow up on and the engineers did this *"off their own bat*". Failure to update specifications with all modifications from the customer caused some problems during testing.

The existence of guidelines ensured that specifications were consistent, and correspondence indicated that the customer was satisfied with the documentation she was receiving. However, one of the engineers stated that:

> *"the specifications were detailed – much more detailed than I have ever seen before, in one way this is a good idea, but sometimes you find problems later because the specification has been taken as gospel".*

One section of the Banking Organizer project suffered from "*feature creep*". Because of this, the engineer working on the software found that changes which "*should be easy but because of this are relatively difficult*".

Implementation

The software development manager implemented the product following procedures which had been written up and experienced very few problems.

Project Management

At the start of the Banking Organizer project, the software development manager produced a table of the project phases and the time it would take to complete each phase. He also considered what personnel were required for the project, taking into account other projects being worked on within the company. Using an automated system, he created a project, showing planned project tasks, responsibilities and duration. Developers were expected to enter actual time spent on the project, allowing the software project manager to track project progress. One software engineer stated that "*project management has improved*" and consequently, the Banking Organizer project "*was a tightly controlled project from the start*". Project progress was updated on a regular basis, as he was now treating the software development group as a "*business unit*". Therefore, emphasis had to be placed on both income from customers and the cost of the group to the company.

Software development group meetings were held at the start of each week. At this meeting, the group reviewed the work done and action items closed during the previous week, the target for the current week, action items open, daily work done by individuals within the group for the previous and current weeks, and any off-site visits to be carried out during the current week. This allowed the group to be updated on the status of projects, and gave a formal forum for discussion around problems that existed on any projects.

Lessons Learned

Following the intervention by the researcher at the beginning of this research project, Computer Craft were to implement six action items based on the top 10 practices which had been identified from the Software Process Matrix as important for the company to improve on. Because a number of key personnel left the organisation left the company soon after these were identified, some of them were not implemented, including Actions 1 and 6.

The second action identified was a combination of setting up procedures to specify and document system requirements and to collect, identify, record and complete new customer requests. This action had been worked on with the development of a specification which included the requirements, functional and technical specification. They also improved their method of dealing with customers, meeting them early on in the development process, updating them on the project and clarifying requests with them. This ultimately effected the test of the product and its implementation. As this had such a positive effect on the processes within the company, it is important that this procedure is maintained for the future, and that the method of dealing with the Banking Organizer project become accepted by the organisation.

Setting up procedures to define delivery contents and to verify all changes to requirements are monitored once the requirements specification was signed off was the third action identified. Two procedures had been written – Installation procedure and Build Release to Customer Services – which covered the first of these. Using these procedures, the implementation of the Banking Organizer went smoothly, with very few problems. While not done formally, changes to the requirements were updated in the specifications when changes were made. This process would need to be better controlled in the future.

Action 4 required that Computer Craft assign a person software quality assurance responsibilities. This had already been done, but their responsibilities were not clear to them. The newly appointed quality assurance engineer, had taken testing as one of her responsibilities, consequently she wrote up detailed test plans and based these on the requirements specification. Her experience was lacking, and it was identified that she needed training to help her fulfill her role within the organisation. Nonetheless, the introduction of detailed test plans had helped the testing of Banking Organizer to be completed efficiently and effectively. A downside to this was that there was little emphasis placed on code reviews within the organisation, and it would be advisable for the company to re-consider this situation. Her other responsibility was the writing and approval of procedures, a number of which were written during the research period. The other practice to be worked on in Action 4 was to define quality criteria and metrics for the project deliverables. This had been partially done, but was not accepted as being an organisation practice.

The fifth action required a procedure to establish product baselines for each product supplied. This was not worked on. It also required that a procedure be set up to assign responsibility for the software development plan, work products and activities. While a procedure was not set up for this, the software development manager took this responsibility on board, and improved the project management within the organisation. This is reflected in the improvements evident in many of the project activities.

Of the ten practices identified, three were not completed during the research period. Another three had been implemented but not formalized with the organisation. It is important that this be done, as the evidence from the Banking Organizer project was that they have aided the improvement of the software process within Computer Craft.

Change in Organisation Processes

In Computer Craft, at the end of the research period, a number of procedures, specifications, guidelines and templates had been written in an effort to streamline the software processes within the organisation. There was still an emphasis on quality assurance and testing of the product. There was no interest shown in the attainment of formal recognition of the process as there was no market requirement for ISO9000 nor any other measurement such as the Capability Maturity Model or SPICE.

Change in Customer Management

Customer management had changed significantly during the research period in Computer Craft. Initially, the software development group had little contact with the customer, and regarded the sales representatives as the customer. In the Banking Organizer project studied at the end of the research, the software developers in Computer Craft had direct contact with the customer.. Comments from the software development group indicated that this way of working contributed to how well the final installation of the product had gone.

During initial research, there had been no procedures nor guidelines used for the collection and documentation of customer requirements. Documentation and collection methods varied between software engineers, even when they were working on the same project. By the end of the research period, a template for the program specification had been introduced, containing a section on requirements. This replaced the previous requirements, functional and technical specifications. Software engineers working on one project were providing and working from consistent information and the customer was satisfied with the output.

The existence and use of the specifications did not prevent "*feature creep*", but contributed to its reduction. Unlike the Data Function product, there were no modifications required after implementation of the Banking Organizer. Also, developers could code the system knowing that very few changes would be made.

Change in Implementation

In the initial stages of the research project, the customer services manager installed software within the customer company, at times involving the software development group. For the project studied at the end of the research period the software development manager carried out the implementation, and no problems were experienced. One of the difficulties faced by the company was that they did not specify what the release criteria for a product was. It had been discussed, and the feeling of one software development meeting, which this researcher attended, was that a release criteria stating the minimum amount that should be contained in a released product was needed. When asked a measure of failures being shipped, the Software Development Manager suggested that "*one failure in every three is bad quality*".

Change in Project Management

The focus of the software development group had changed throughout the research period. Schedules were set in conjunction with the customer rather than being dictated by the customer services group. Engineers were expected to give the manager feedback, and thus, he was able to keep a tighter control on the project.

Analysis of Change in Computer Craft

Overall, there many changes were made in Computer Craft during the research period. Probably the most significant of these was the manner in which the customer was dealt with. The software development manager worked closely with the project manager in the client company, and the group produced specifications which contained the customer requirements. Changes to these were discussed with the software engineer, modifications were normally made to specifications and "feature creep" was not prevalent on the project. This in turn improved both the testing and implementation of the product.

The implementation of the practices as identified when using the Software Process Matrix was partially responsible for the improvements in Computer Craft's software process. However, without management support for the project, particularly when the company was going through recruitment difficulties, the identification of the practices alone was not sufficient. It was important that the exercise was followed through and the practices were implemented within the organisation. The software development manager recognized that changes were required and used the Software Process Matrix as a basis for identification of the most relevant changes.

The company has not identified any market requirement for the implementation of any particular standards such as ISO9000 or for being assessed using SPICE or the Capability Maturity Model. It is possible that customers may look for this in the future, and the company is now better placed for proceeding with such actions.

Authors

Ita Richardson received a B.Sc. in Applied Maths from NIHE, Limerick, in 1983, an M.Sc. in Maths and Computing from the University of Limerick, in 1993. She has held the CPIM from the American Production and Inventory Control Society since 1986 and was awarded a Certified Diploma in Accounting and Finance from the Chartered Association of Certified Accountants in 1993. She is also a member of the B.C.S and a Chartered Engineer.

She worked with Wang Laboratories, B.V., Limerick for 8 years until 1991, where she was mainly involved in Programming and Systems Analysis of systems used in the manufacturing plant production, Materials (including MRP), Distribution, Financial, and Personnel. In latter years, she also had responsibility for the maintenance of Systems Standards, and training of Programmers and Analysts.

Her M.Sc. project was a Simulation of the Automatic Storage and Retrieval System used in the manufacturing plant. Her current research involves the investigation of Quality Tools, particularly Quality Function Deployment, and how they can be applied to the software development process.

She was appointed to lecturer in 1998. Her PhD research is in the application of Manufacturing Quality Techniques to Software Process Improvement in Small Software Development Companies. She is a founder-member of the Small Firms Research Unit at the University of Limerick, whose research is specifically concerned with the growth and development of small firms.

Kevin Ryan received a B.A.I. in Engineering in 1971 and a Ph.D. in Computer Science in 1978 all from Trinity College Dublin. His Ph.D. was concerned with the simulation and optimisation of bus scheduling operations. He lectured in Computer Science at TCD in 1972 before working as programmer training manager for RCM Ltd. Zambia where he recruited and trained the first Zambian programmers. In 1979 he was on the faculty of the University of Kansas before rejoining Trinity College Dublin in 1980. He spent 1985/86 as a guest researcher at Linkoping University. In 1990 he became Professor and head of the Department of Computer Science and Information Systems and, since September 1994, has been Dean of the College of Informatics and Electronics.

He has published papers on simulation, programmer training, software methods, expert systems, natural language processing and requirements engineering. His current research interests include systems design methods, knowledge-based software development is currently involved in the EU funded network of excellence in Requirements Engineering (RENOIR).

He has acted as Evaluator for the EU and as external examiner for universities in Ireland and the UK.

He has lectured on Programming Languages, Program Design Methods, Business Computing, Data Structures, Operating Systems, Social impacts of Computing, Software Engineering and Artificial Intelligence.

Computer Craft Ltd.

Computer Craft Ltd. was established in April, 1984 to develop software for use in specific business functions. The original entrepreneur, himself a software engineer, is the Group Managing Director and is no longer involved in the development of software. The company, based in the mid-west region of Ireland, employs 16 people in the Irish office and in a sales office in the U.K. Of these, the software development group of 4 people is based in Ireland.

References

[1] Peterson, Bill, Transitioning the CMM into Practice in *Proceedings of SPI 95 - The European Conference on Software Process Improvement, The European Experience in a World Context*, 30th Nov-1st Dec, 1995 Barcelona, Spain, pp. 103-123.

[2] Draper, Lee, Kromer, Dana, Moglilensky, Judah, Pandelios, George, Pettengill, Nate, Sigmund, Gary, Quinn, David Use of the Software Engineering Institute Capability Maturity Model in Software Process Appraisals, output from *CMM v2 Workshop*, February, 1995 Pittsburgh, Pennsylvania, U.S.A.

[3] Peterson, Bill, Software Engineering Institute, *Software Process - Improvement and Practice*, Pilot Issue, August, pp 68-70, 1995.

[4] Debou, Christophe, Kuntzmann-Combelles, Annie, Linking Software Process Improvement to Business Strategies: Experiences from Industry in *Proceedings of SPI 98, The European Conference on Software Process Improvement*, 1-4th December, 1998, Monte Carlo.

Recommended Reading

Further reading on Quality Function Deployment:

[1] Clausing, Donald, *Total Quality Development*, ASME Press, U.S.A., 1994.

[2] Cohen, Lou, *Quality Function Deployment, How to Make QFD Work for You*, Addison-Wesley, U.S.A., 1995

[3] ReVelle, Jack B., Moran, John W., Cox, Charles A., *The QFD Handbook*, John Wiley & Sons Inc., U.S.A., 1998.

Further reading on the Software Process Matrix:

[1] Richardson, Ita, Quality Function Deployment - A Software Process Tool? in *Proceedings of the Third Annual International QFD Symposium*, Linkoping University, 1st-2nd October, 1997, Linkoping, Sweden, pp 39-49, Volume 2.

[2] Richardson, Ita, Using Quality Function Deployment to Develop Action Plans for Software Process Improvement, *Proceedings of the 10th Software Engineering Process Group Conference (SEPG '98)*, 9th-12th March, 1998, Chicago, U.S.A.

Improving an Evolutionary Development Process – A Case Study

Erik Arisholm
Univ. of Oslo, erika@ifi.uio.no

Jon Skandsen
Genera AS, jsk@genera.no

Knut Sagli
Genera AS, ksa@genera.no

Dag I.K. Sjøberg
Univ. of Oslo, dagsj@ifi.uio.no

Abstract

Genera AS is a vendor of a CASE tool called Genova. The work described in this paper aims to define and evaluate an evolutionary development process (the Genova process) to complement and support the use of the Genova tool. As a starting point, we used a lightweight version of the Rational Unified Process in a development project for one of our customers. This process was instrumented to enable process improvement activities. An essential question is what to improve – the defined or the actual process? Based on quantitative and qualitative data, we identified improvements related to the distribution of test effort throughout the life cycle. Furthermore, we gained useful experience on the management of evolutionary development projects. Our case study also provided insight for instrumentation of the process for collecting data related to the cost of changes.

1 Introduction

Evolutionary development has been proposed as an efficient way to deal with risks such as new technology and imprecise or changing requirements (Boehm 1988). The main idea is to resolve risks early by incrementally evolving the system towards completion instead of relying on the traditional "big-bang" waterfall approach (Royce 1970). Thus, an important objective of evolutionary development is to identify the "real" needs of the customer as the system evolves. While experience reports show some success in the application of evolutionary development (Gilb 1988, Zamperoni *et al.* 1995), there are unfortunately few empirically based guidelines on how to apply such processes in different development contexts. The Rational Unified Process is being adopted by a growing number of small- and medium-sized software development companies. However, we believe that the number of prescribed deliverables, roles and activities are perhaps too many to be practical for use in relatively small development projects.

Genera AS is a vendor of Genova, which is an advanced CASE tool for object-oriented analysis and design, dialog modeling, and automatic application generation and database generation (Arisholm *et al.* 1998). In conjunction with the development of the Genova *tool*, we are also developing the Genova *process*. The goals of the work presented in this paper are to:

- Define the Genova process as a scaled down version of the Rational Unified Process. We believe that such a "light-weight" process will increase the likelihood that the process is followed in smaller development projects.
- Gain practical experience with the Genova process in real development projects.
- Instrument the process to determine a process performance baseline from which process improvement activities can be performed.

The Genova process was evaluated while it was used in a development project run by Genera AS for the Norwegian airline, Braathens. In this development project, we were able to deliver a software system, within the agreed schedule, that matched the customer needs reasonably well. At present, the system has been operational for seven months with only minor post-delivery adjustments. Thus, this case study provides one instance of an evolutionary development project that succeeded. However, our experience indicates that it may be difficult to provide a defined process that is followed in real life. In this case study, testing was performed too late in comparison with the prescribed process. Although it is, in retrospect, uncertain whether this lack of process conformance could have been avoided by the development team, it is likely that it contributed to many costly last-minute changes to the software. Our case study has also provided insights for quantitative evaluation of process improvement activities; a data collection process for analyzing the cost of implementing changes to the software is described.

The remainder of this paper is organized as follows. Section 2 elaborates the goals and criteria for defining the Genova process and gives an overview of the process. Section 3 describes the case study conducted to evaluate the process in an industrial setting. Empirical results from the case study are provided in Section 4. Section 5 discusses implications of the case study and suggests process improvements. Section 6 describes on-going and future work. Section 7 concludes.

2 The Genova Process

The aim of our work is to provide a defined process (the Genova process) for use both in internal product development and in conjunction with external development projects of Genera AS. Furthermore, the process is also intended to provide methodical guidelines with specific support for the Genova tool (Arisholm *et al*. 1998).

Criteria for Defining the Genova Process

Initially, we considered Gilb's EVO (Gilb 1988), HP Evolutionary Fusion (Cotton 1996), Dynamic Systems Development Method (DSDM) and Rational Unified Process (RUP) (Kruchten & Royce 1996) as candidates for the Genova process. These processes differ substantially in their prescribed roles, schedules, activities and deliverables. However, it is unclear under what circumstances one approach is more suitable than the alternative approaches. To our knowledge, no empirical studies exist that compare the strengths and weaknesses of these processes. Without such a scientific foundation, the decision to base the Genova process on RUP was mainly motivated by the growing popularity of this process in industry; there is a competitive need to be compatible with the terminology and the deliverables of this emerging "de-facto" industry-standard in our market segment.

One advantage of using RUP terminology and deliverables (e.g. "elaboration" and "use case model") is that this terminology may be more commonly known among developers in industry. However, we believe the number of prescribed deliverables, roles and activities in RUP are too many – at least for practical use in small and medium-sized development projects. Thus, our aim is to define a smaller development process, based on RUP, which is better suited for small development projects. We believe that such a "light-weight RUP" will increase the likelihood that the process actually is understood and consequently followed. This issue is also known as "process conformance", which has been defined as "The degree of agreement between a process execution and a process model" (Sørumgård 1997).

Process Description

Fig. EAJSKSDS.1 depicts the evolutionary delivery of increments, prescribed by the Genova process. The process prescribes the delivery of an initial architectural baseline and a high-level design. Each increment is developed by iteration of all major process activities, including analysis, design, coding and test.

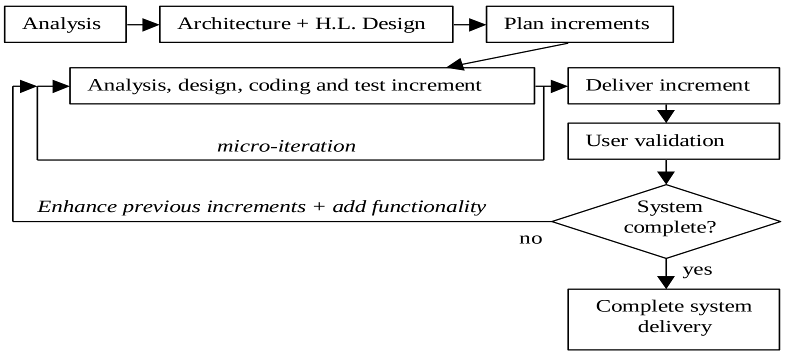

Analysis
Architecture + H.L. Design
Plan increments
Analysis, design, coding and test increment
Deliver increment
micro-iteration
User validation
Enhance previous increments + add functionality
System complete?
no
yes
Complete system delivery

Fig. EAJSKSDS.1: The Genova evolutionary delivery life cycle

There may be up to three iterations per increment:

Iteration 1: Implement the most important functional requirements of the increment. Iteration 1 serves as an evaluation of the design of the increment.

Iteration 2: Implement the remaining functional requirements of the increment. Enhance functionality developed in iteration 1.

Iteration 3: Stabilize increment.

The increment is then delivered to end-users for evaluation. The next increment will contain enhancements to the previous increments as well as new functionality. The system delivery is completed when all functionality has been delivered and no further enhancements are required. Table EAJSKSDS.1 provides an overview of the roles, activities and resulting artifacts prescribed by the Genova process.

Table EAJSKSDS.1: Roles, Activities and Artifacts of the Genova process

Role	Activity	Artifact
Project initiator	Develop vision	Vision statement
System analyst Project initiator	Find actors and use cases	High level use case model
System analyst	Detail use cases	Detailed use case model
Project leader	Plan project evolution	Project plan (iterations and increments included)
System analyst Architect Project leader	Risk management (prioritize use cases)	Project plan
System analyst	Develop sequence diagram	Class candidate/operation list
System analyst	Find class candidates	Class candidate list
Designer Architect	Develop domain model	Domain model (class model)
Customer Developer Visual designer	Dialog modeling	Application visual interface Source code Class candidate/operation list
Architect	Develop architecture	Architecture document
Designer	Develop design model	Design
Developer	Generate code	Class/operation skeletons
Developer	Code	Application source code
Build responsible	Build code	Components, Binaries
Test responsible	Develop test plan	Test plan
Tester Customer Developer	Test	Test report Improved code

3 Industrial Case at Braathens

The Genova process was used in a development project for the Norwegian airline Braathens. The development team consisted of from 2 to 6 developers and one experienced project manager. The system being studied implemented an automated customer service for Braathens' frequent flyer program, "Wings". The system is a three-tier application consisting of Java/HTML clients, a middle-tier component for transaction processing and information retrieval, and a mainframe database server. The middle-tier module was implemented as classes in Visual Basic 6 and bundled in ActiveX components running on a Microsoft Transaction Server. After week 22, the system became operational. Three increments were delivered during these 22 weeks: at week 6, 11 and 22, respectively.

Process Instrumentation

The process was instrumented with a small number of process and product measures. The purpose of the process instrumentation was to:

1. Establish a quantitative baseline for process improvement activities.
2. Provide a means to assess process conformance.
3. Evaluate the effect of improvement activities.

During the Braathens project, the process instrumentation was used to establish a quantitative baseline (1) and to assess process conformance (2). Improvement activities were not performed in conjunction with the Braathens project. Hence, at present, effect measurements (3) have not yet been conducted.

Weekly effort data in person-hours was recorded by each team member for important activities (analysis, design, code, test and administration) to implement the system. Coding effort data was reported individually for each module of the system. In addition, internal product measures were collected. Using the configuration management tool and a code parser for Visual Basic, product measures were collected from the middle-tier module based on weekly versions of the software throughout the 22-week period. The internal product measures consisted of, among others, module size and class size (in SLOC), number of classes, number of methods per class and coupling between classes. The internal product measures connected to the coding effort data enabled us to compute "coding productivity" for the middle-tier module. The internal product measures were also intended to provide quantitative indicators for product quality assessment. However, this topic is beyond the scope of this paper. Further details of the product quality assessment are described in (Arisholm & Sjøberg 1999a, Arisholm & Sjøberg 1999b).

4 Case Study Results

This section reports the results from the Braathens case study. The summary data in Table EAJSKSDS.2 indicates that, during the development of the middle-tier module, there was a significant amount of rework. This rework is indicated by the ratio between net productivity and gross productivity for implementation of this module. For example, during the five weeks of the second increment, 3191 source lines of code (SLOC) were added or deleted from the module but the module grew by only 1128 SLOC. Only about 35% of the total amount of coding on the module contributed to increased module size (yielding a *rework ratio* indicator of 65% for the second increment). Note, however, that the SLOC-based productivity and rework measures may be of questionable validity (Arisholm and Sjøberg 1999b).

Table EAJSKSDS.2: Summary process data for the middle-tier module

Process/product measure	Incr.1	Incr.2	Incr.3
	(week 1-6)	(week 7-11)	(week 12-22)
Coding effort per incr. (person-hours)	149	209	517
Changes per incr. (SLOC added + deleted)	1120	3191	5203
Gross productivity (SLOC added + deleted per hour)	7.5	15.3	10.1
Net productivity (SLOC growth per hour)	4.6	5.4	4.8
Rework ratio (% of effort not contributing to code growth)	39%	65%	52%
System size (SLOC)	687	1815	4307

Fig. EAJSKSDS.2 depicts the distribution of effort (in person-hours) for various process activities (analysis, design, coding[1], documentation/test, administration and installation) during the 22 weeks. While there are some overlap in the process activities, there is still a somewhat "phased" distribution of activities over time – to some extent resembling that of the traditional waterfall development process. For example, formal testing was only conducted in the third increment and not in the first two increments as prescribed in the Genova process. *Informal* testing was done by each developer throughout the coding activity. However, the separate, *formal* test activity including, for example, deployment in a dedicated test environment, writing test cases and applying test-logging tools, was not initiated before towards the end of the last increment.

[1] The coding effort in Fig. EAJSKSDS.2 reports the total coding effort for all modules in the system, and not only for the middle-tier module as in Table EAJSKSDS.2.

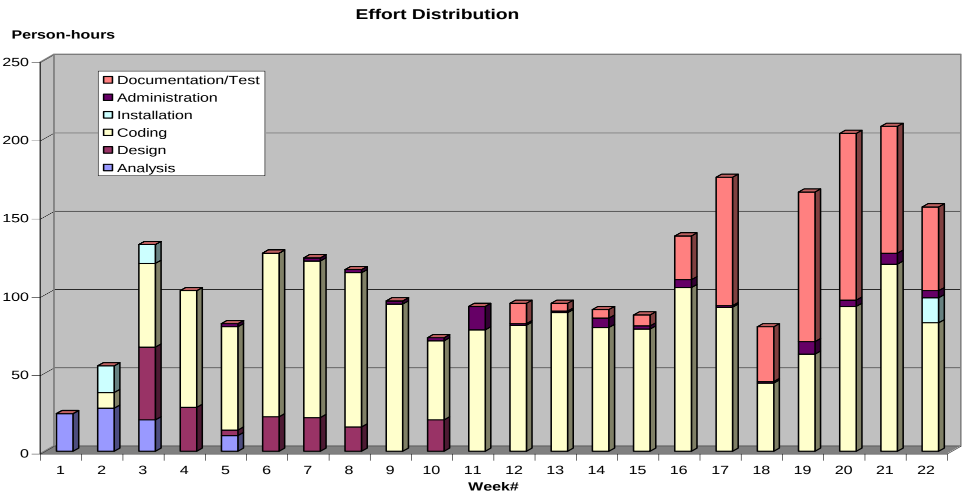


Fig. EAJSKSDS.2: Effort distribution for activities during the Braathens project

5 Discussion

In this section, we discuss the results reported in Section 4, with emphasis on rework and process conformance aspects.

Rework

Interviews with the development team indicate that the amount of rework experienced on the development project may in part be explained by uncertainties caused by the new technology used in the project. In particular, there was a mismatch between the promised and actual quality of certain development tools and libraries. A significant amount of coding effort was spent on trying alternative "work-around" solutions to compensate for flaws in the development tools and libraries. The rework is probably also a result of the evolutionary and incremental way in which the module was developed.

A certain amount of rework is a natural part of evolutionary development – it is necessary in order to produce a good product. However, too much rework may result in unacceptably low productivity and unnecessary costs. For rework to be a useful and valid process performance indicator, it should be balanced with product quality indicators, such as customer satisfaction, the number of change requests from users after system delivery, etc. This balanced view of initial rework versus customer satisfaction and later change requests may provide a meaningful baseline for improvement activities in future evolutionary development projects.

Process Conformance

Ensuring process conformance is important for several reasons (Sørumgård 1997):

- To ensure a stable process execution, that is, achieving a predictable process.
- To ensure the validity of the data, information, experiences and knowledge that are acquired throughout the development projects.

Without a "reasonable" degree of process conformance, it may be difficult to determine the effect of process improvement activities. A key question is *what* to improve – the defined or the actual process? What are the reasons for reduced process conformance: is a lack of process conformance due to a flaw in the defined process or is it due to a flaw in the executed process? What are the implications of process conformance on software process improvement: does it make sense to "improve" a defined process that has not been followed?

In the Braathens case study, testing was not performed as prescribed by the defined process. Interviews with the developers indicate that the delayed testing contributed to many costly last-minute changes to the software. For example, many of the detailed requirements of the client-tier of the application were not discovered before during the detailed write-up of the test cases, resulting in late rework. This provides a partial explanation of the large volume of coding effort towards the end of the final increment (Fig. EAJSKSDS.2).

One explanation for this lack of process conformance was that the initiation and execution of the Genova process at Braathens were quite informal. However, insufficient guidelines for initiation and execution of the testing activities may also have contributed to this "flaw". In the Braathens case, one problem was that machine resources for deployment and load testing were made available very late by the software customer. Thus, it is uncertain whether the resulting lack of process conformance could have been avoided by the development team. Both the software vendor and the customer would have benefited if test facilities had been made available from the outset of the project, allowing early testing according to the prescribed evolutionary life cycle. This aspect should have been addressed explicitly in the initial contract between the software vendor and customer. The result of this experience is thus a suggestion for improvement of one aspect of the defined Genova process: contractual guidelines regarding test facilities should be incorporated in the process description.

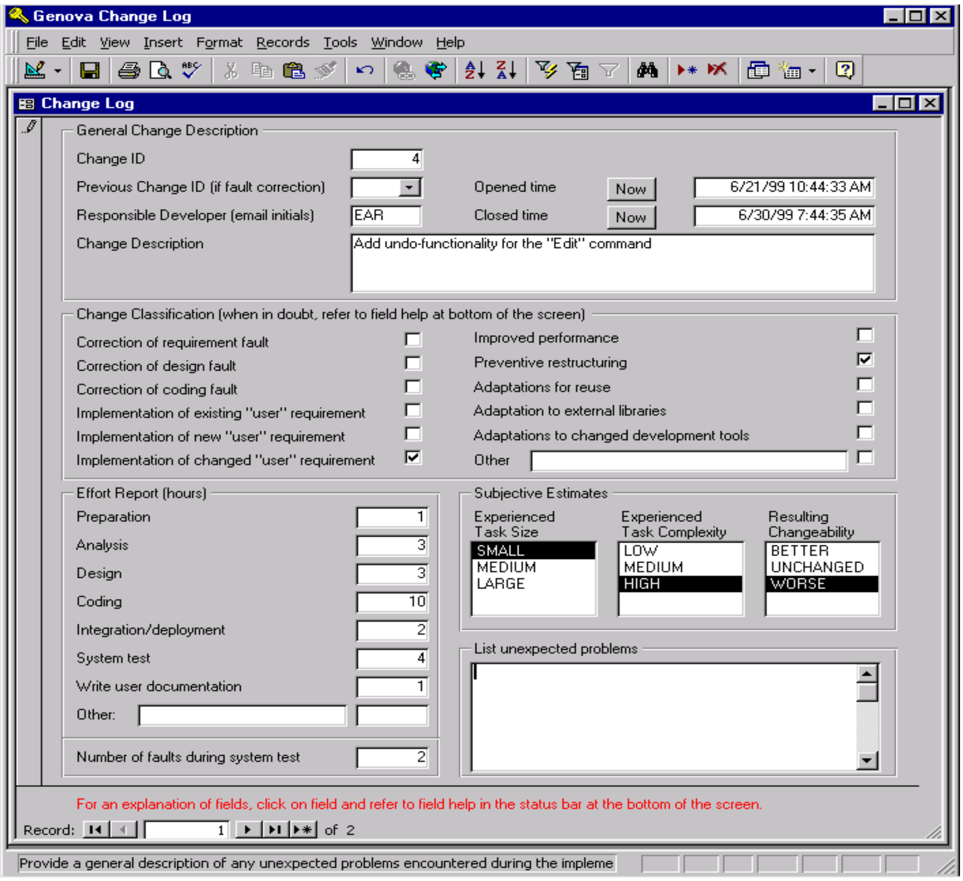
6 Future Work

Further evaluation and improvement of the Genova process are currently in progress. A new case study has been initiated in conjunction with an internal product development project (of the Genova tool itself) in Genera AS. The programming languages used are C++ and Java. This development project is larger than the Braathens case, consisting of about nine developers. The project is expected to last for several years, providing good opportunities for process improvement activities including effect measurements.

Based on the experiences from the Braathens case study, we have refined the process instrumentation. The data collection process has been defined to ensure that the developers

1. classify all changes and assign a change ID,
2. tag each file-level check-in with the correct change ID, and
3. report process data (change effort, subjective change complexity, number of discovered faults, etc.) per change.

A data logging tool has been implemented to support this process (Fig. EAJSKSDS.3). At present, 34 changes to the Genova CASE tool have been recorded by the developers using the Genova change logger tool. The preliminary results are very promising. The developers have understood the potential long-term benefits of using the tool. Therefore, they accept the extra overhead incurred for the data reporting, and do indeed use the logger tool. Each individual change reported in the log can be traced in the source code using the configuration management database. This traceability allows us to collect internal product measures related to each change (using C++ and Java code parsers), coupled to external indicators such as change effort and defect data. We believe that we now are in a good position to conduct process improvement effect measurement by analyzing trends in the costs and consequences of implementing changes to the software. Pending further evaluation, our long-term goal is to implement this data collection process in all internal product development projects at Genera AS.



General Change Description

Change ID: 4

Previous Change ID (if fault correction):

Responsible Developer (email initials): EAR

Opened time [Now] 6/21/99 10:44:33 AM

Closed time [Now] 6/30/99 7:44:35 AM

Change Description: Add undo-functionality for the "Edit" command

Change Classification (when in doubt, refer to field help at bottom of the screen)

☐ Correction of requirement fault
☐ Correction of design fault
☐ Correction of coding fault
☐ Implementation of existing "user" requirement
☐ Implementation of new "user" requirement
☑ Implementation of changed "user" requirement
☐ Improved performance
☑ Preventive restructuring
☐ Adaptations for reuse
☐ Adaptation to external libraries
☐ Adaptations to changed development tools
☐ Other

Effort Report (hours)

Preparation	1
Analysis	3
Design	3
Coding	10
Integration/deployment	2
System test	4
Write user documentation	1
Other:	
Number of faults during system test	2

Subjective Estimates

Experienced Task Size: [✓ SMALL] MEDIUM LARGE

Experienced Task Complexity: LOW MEDIUM [✓ HIGH]

Resulting Changeability: BETTER UNCHANGED [✓ WORSE]

List unexpected problems

For an explanation of fields, click on field and refer to field help in the status bar at the bottom of the screen.

Record: 1 of 2

Provide a general description of any unexpected problems encountered during the impleme

Fig. EAJSKSDS.3: User interface of the change logger tool

7 Conclusions

An important objective of evolutionary development is to identify the "real" needs of the customer early, hence achieving improved customer satisfaction and avoiding expensive last-minute rework. In this paper we gave an overview of the first version of the evolutionary Genova process. A preliminary evaluation of the Genova process was conducted in an industrial development project at Braathens in Norway. The case study provided one instance of an evolutionary development project that succeeded. However, based on quantitative and qualitative data, we identified improvements related to the distribution of test effort: the late initiation of formal testing contributed to unnecessary rework. We believe that less rework would have been required if formal testing had been conducted in each increment according to the prescribed process. Thus, more accurate contractual guidelines will be incorporated in the process description to ensure better process conformance for the test activity in future development projects. The effect of the suggested process changes still needs to be evaluated. Such effect measurement will use a new change logger tool for empirical assessment of the cost of implementing changes.

Acknowledgements

We acknowledge constructive comments from the anonymous reviewer. We are grateful for the support from Erik Amundrud, Bjørg Håland and Stein Grimstad from Genera AS. We thank Kari Juul Wedde and Tor Stålhane from SINTEF for useful assistance regarding GQM. The research project is funded by The Research Council of Norway through the industry-project SPIQ (Software Process Improvement for better Quality).

References

Arisholm, E., Benestad, H.C., Skandsen, J. and Fredhall, H. (1998). “Incorporating Rapid User Interface Prototyping in Object-Oriented Analysis and Design with Genova”. In: *Proceedings of NWPER'98 Nordic Workshop on Programming Environment Research,* Sweden 1998, pp. 155-161

Arisholm, E. & Sjøberg, D.I.K: (1999a). “Empirical Assessment of Changeability Decay in Object-Oriented Software”. In: *ICSE'99 Workshop on Empirical Studies of Software Development and Evolution,* Los Angeles, CA, pp. 62-69

Arisholm, E. & Sjøberg, D.I.K. (1999b). “Towards a Framework for Empirical Assessment of Changeability Decay”. Accepted for publication in *Journal of Systems and Software, 1999.*

Boehm, B.W. (1988). “A spiral model of software development and enhancement.”. *IEEE Computer*, Vol. 21, No. 5, pp. 61-72.

Cotton, T. (1996). “Evolutionary Fusion: A Customer-Oriented Incremental Life cycle for Fusion”. *Hewlett-Packard Journal,* Vol. , No. , pp.

Gilb, T. (1988). *Principles of Software Engineering Management*. Addison-Wesley.

Kruchten, P. & Royce, W. (1996). “A Rational Development Process”. *CrossTalk,* Vol. 9, No. 7, pp. 11-16.

Royce, W. (1970). “Managing the development of large software systems: Concepts and techniques.”. In: *Proceedings of IEEE WESTCON*, Los Angeles 1970, pp. 1-9

Sørumgård, L.S. (1997). Verification of Process Conformance in Empirical Studies of Software Development. PhD Thesis, NTNU.

Zamperoni, A., Gerritsen, B. and Bril, B. (1995). Evolutionary Software Development: An experience Report on Technical and Strategic Requirements. Technical Report TR-95-25, Leiden University, The Netherlands.

Rightsizing versus client server with an outsourcing approach: a case study of a big public service organisation

Ing. Francesco Marinuzzi, Ph.D.
C.I.M. Consultancy on Information systems and Methodologies
Via A. Manno, 11 - 00133 Rome Italy
Email: Francesco@Marinuzzi.com
Web: http://www.marinuzzi.com

Key Words

Software Engineering, Rightsizing, Downsizing, Outsourcing, Software Performance Engineering, Downsizing.

Introduction

This paper presents a case of rightsizing, with an outsourcing approach, of a mainframe based information system.

A full downsizing process, is a highly complex process due to the following reasons:

- The need to manage, at the same time, the old and the new technology and environment for the parallel periods;
- The need to migrate in the new platform the millions of LOC (line of code) of the several applications.

In this paper we describe how that process can be performed in an outsourcing framework. We discuss which are the critical factors that assure an efficient process and big savings from the cost/benefit and cost/performance point of view.

There are several key factors to be considered in order minimizing the risks of failure and maximizing the success.

The paper is organized in the following three parts:

- The Rightsizing process;
- The case study and the results obtained;
- The lessons learned from the experiences done.

The rightsizing

Generally we can distinguish two main streams of the Rightsizing process: the downsizing and upsizing processes.

The downsizing process is characterized by data and process shifting from Mainframe to desktop connected with LAN and WAN network.

The following steps, instead, characterize the Upsizing process:

- The integration and connection of stand alone workstations or LAN
- The development of distributed applications on this new architecture

The Rightsizing of the applications and systems, that is their Downsizing or Upsizing, provides a major opportunity for cost savings and improving the flexibility of the information systems.

In the following figure we see the rightsizing of the case study. It is from a traditional architecture based on a MVS IBM to a new distributed Client-Server architecture. The new Client-Server applications are executed on several application servers.

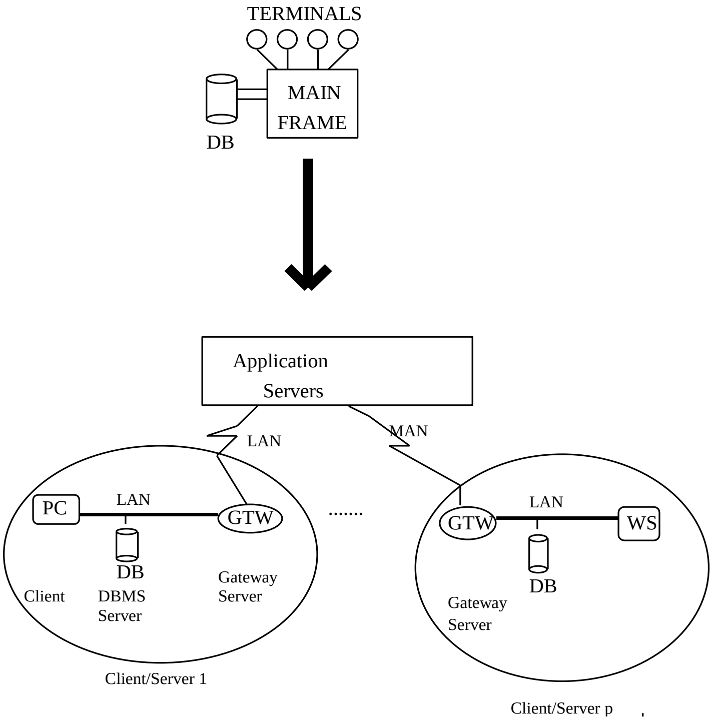


Figure 1. Rightsizing of a centralized architecture

Moving to Open and Client-Server System environments allows organizations to take advantage of several opportunities: the new cost/performance relations for the hardware components, the 'easy to use' graphical interfaces, the portability of the software, the adoption of faster software life cycles like RAD, the use of the information highways (for example internet) and so on [4].

Most of the organizations invest in rightsizing processes to build Client/Server architectures.

Actually, upsizing and downsizing process has become a phenomenon of big importance for a lot of organizations. It is possible to preview that the next years the rightsizing tendency will increase very much, and will become a rule.

Anyway we must consider, from the point of view of the architectural structure, that a rightsizing process is complex and may imply an increase of complexity of the final system if it is not well planned.

A typical distributed Client-Server architecture is characterized by the presence of workstations and personal computers acting as clients, by the interconnection of different groups of these computers through different local and wide networks, by the presence of several servers including eventually a mainframe as a file server. It is typical of the local network that there is a local database; the local stations of the network are able to reach this database instead of the remote database.

In order to lower the complexity of the process, an outsourcing of all the IT activities, for the rightsizing period, or even more, can prove effective. The IT system after the outsourcing/rightsizing period (typically from 3 to 5 years) becomes technologically updated and easier to maintain (especially if the outsourcing/rightsizing contract provides strict guidelines for the development of the new software and for the Reverse Engineering projects).

The case study

The starting environment

The company is one big public organization providing services to the Italian citizens.

The environment is characterized by a mainframe with MVS as operating system, DB2 as DBMS and 10 million of Line of Code mainly in Cobol for a total of more then 40.000 FP.

The mainframe provides services to 500 clients connected to a fast Ethernet LAN. The Clients execute terminal emulation software to execute the MVS applications.

The LAN is in a Windows NT environment with over than 10 servers and a SQL server as DBMS for stand alone applications developed in visual languages. The LAN has some parts in optical fiber (ATM 155 Mbs) for a MAN.

The level of integration between the two environments is low and from the organizational point of view there are two separate teams with different skills and competences.

The process

The main phases of the process have been the following:

- Feasibility study;
- European bid;
- Start up & Monitoring;

The feasibility study has identified the system and user requirement, and has evaluated several alternatives for a cost/benefit analysis.

The bid has been awarded using the “more fauvorable” economical offer rule instead of the minimum price to maximize the quality: the economical starting value was about 50 mm US$.

The start-up lasts a period of 9 month during which the winner mainly had to assure the essential services.

After the start-up period, for all the types of activity there is a monitoring and service level measurement activity in line with the contract papers.

The requirements

Activity

The main activities of the bid have been:

- The migration to open system of the ISC (information Service Centre).
- The outsourcing of the most of IT activities.

Service level and Quality

Each participant to the tender had to present several Quality and Project Plans (for each type of activity as development, maintenance, innovation, conduction, etc.) and fill several service levels tables with the values they wanted to assure under economical penalty.

To assure the best software productivity they had, also, to fill a productivity table (in function point) with 27 cells for all the size, risk category and environments.

Innovations projects

The main requirement was to move to an “open system” not necessarily getting rid of the host. The new software had to be portable: easy to migrate at least in one other environment.

Software documentation

To assure the preservation of all the functionalities of the existing software, the participants had to present a detailed Reverse Engineering Plan for each Application Area.

Performance

From experience the performance requirements are particularly critical in the rightsizing processes. For this reason we will first describe the problem and the then we will present the followed approach.

The problem of the performance requirements

Performance evaluation of a classical concentrated architecture focus mainly on the data access while in a distributed architecture like a Client-Server system, the main problems come especially from software bottlenecks, contention on resources, locking and congestion of the networks (LAN and WAN).

From the modelling point of view a Client/Server model must take into account the synchronous inter-task communication associated with client/server layered architecture.

The model must foresee the possibility of a server that can act also as a client requiring service to another server (active server), and so on: this double behavior renders the “software bottleneck” possible and a peculiarity of the Client-Server systems [6], [7].

Quantitative performance prediction models are vital in order to obtain efficient applications and systems without increasing the time or the cost of the projects due to necessary remaking.

There are several approaches to manage the performance requirements when downsizing existing applications. The analytical ones is more effective in the first phases and the simulative one in the last ones of the software life cycle.

The analytical approach has several advantages for its low computational cost and the richness of the results and predictions that can be obtained since the first phases of the developments for what-if studies [1], [2].

In the rightsizing processes, at least in the first phases of the system design, a simulative approach is not effective because it would require the values for many system parameters that would not be available and would require a great amount of time to solve the several possible scenarios [3], [5].

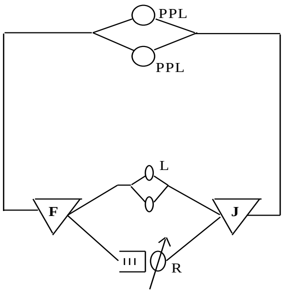
Furthermore, it is also very risky to follow a “Fix-It-Later” approach: it consists of the developing of the applications not caring the performance requirements except at the end of the realization just before their delivery. The risks are a low service level of the applications and higher costs and more time for a later remaking.

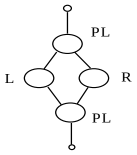
Another difficulty arises from the fact that in Client/Server systems the performances and thus throughput and response time are dependent not only on data accesses time like in classical mainframe oriented applications, but also on communications delays due to congestion, distributed locking and software layer architecture: this renders a lot of tool for the performance prediction focused on the data (for example Windtunnel of Bachman Inc.) or on the hardware obsolete. For further information it can be seen [1], [6].

The markovian theory is very effective to evaluate several “what if” scenarios before the downsizing of the applications and their partition between the client and the server part. The analytical “product form” models based on the markovian queue theory are much easier to be solved respect the simulation models that often require too many details and use several hours of CPU time for one scenarios.

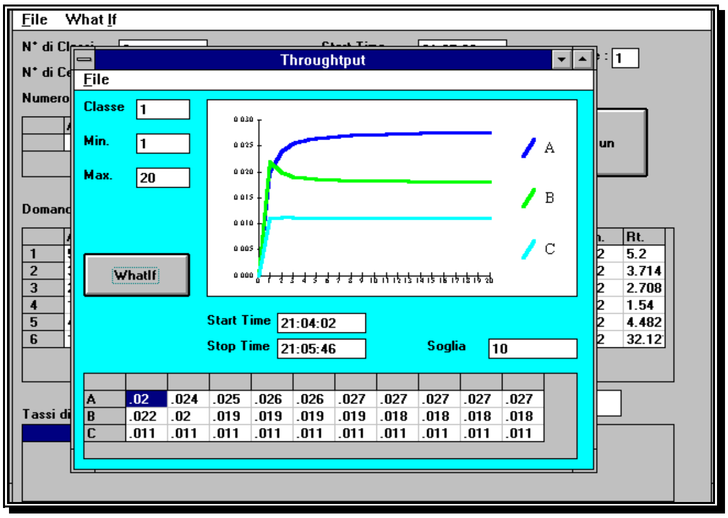
The main reason is that the “product form” models are specific analytical models that do not require the solution of the whole model to compute required performance

indexes. They can be applied to the specific case successfully without any need of simulation or “not product form” models. In fact their solutions will be used mainly to support the software architects to decide how to partition, statically or dynamically (for example with java) the application data and logic in the several software horizontal (client and servers) and vertical parts (components, middleware): this will avoid the “software bottlenecks” typical of a lot of client server systems.
In the following figure we can see a typical quantitative queue prediction model; each PPL and L delay represents a local Client-Server system and is particularly considered the concurrency between the local and remote elaborations in the derivation of the Response time





This model is useful, for example, to see how the performance indexes change with the internal server application “degree of parallelism”. In the following figures the throughput, at the user level, for each class of the system for different values of the internal server application “degree of parallelism”.



Classe 1

Min. 1

Max. 20

Start Time 21:04:02

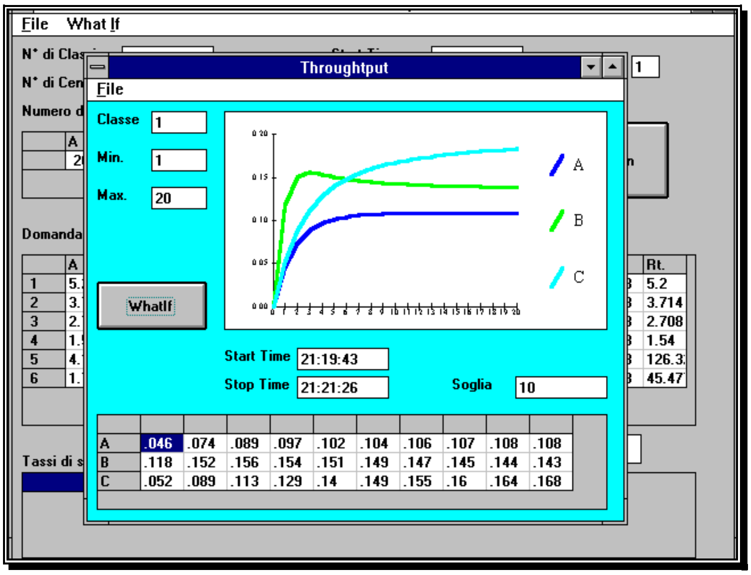
Stop Time 21:05:46

Soglia 10

A	.02	.024	.025	.026	.026	.027	.027	.027	.027	.027
B	.022	.02	.019	.019	.019	.019	.018	.018	.018	.018
C	.011	.011	.011	.011	.011	.011	.011	.011	.011	.011

Throughput for a “degree of parallelism” of the application software server equal to 1.

As we can see in the figure the throughput saturates with the increase of the number of users and has low values: that means there is a bottleneck at the user level (***software bottleneck***).



Classe 1

Min. 1

Max. 20

Start Time 21:19:43

Stop Time 21:21:26

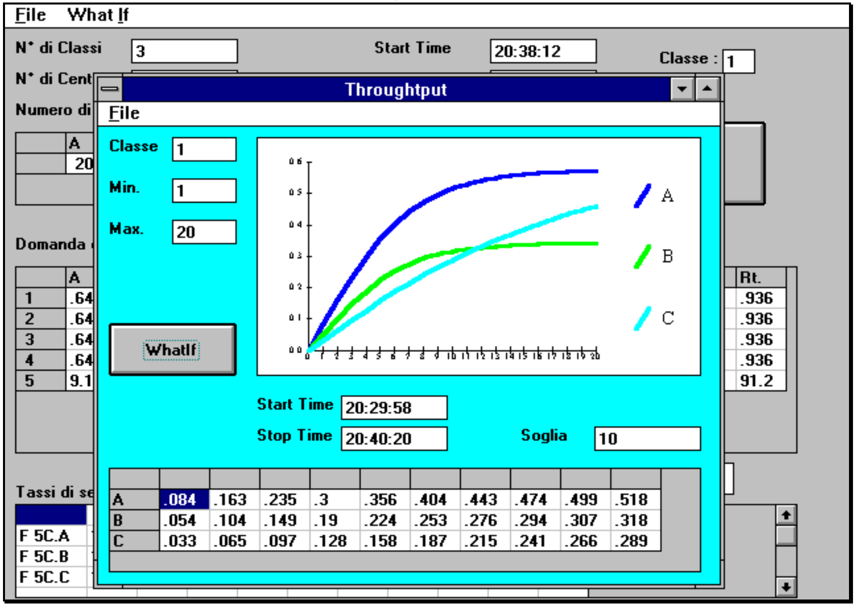
Soglia 10

A	.046	.074	.089	.097	.102	.104	.106	.107	.108	.108
B	.118	.152	.156	.154	.151	.149	.147	.145	.144	.143
C	.052	.089	.113	.129	.14	.149	.155	.16	.164	.168

Rt.
5.2
3.714
2.708
1.54
126.3
45.47

Throughput for a “degree of parallelism” of the application software server equal to 20.

As we can see in the figure above throughput saturates in a smoother way with the increase of the number of users and has higher values than before.



N° di Classi 3

Start Time 20:38:12

Classe : 1

Classe 1

Min. 1

Max. 20

Start Time 20:29:58

Stop Time 20:40:20

Soglia 10

A	.084	.163	.235	.3	.356	.404	.443	.474	.499	.518
B	.054	.104	.149	.19	.224	.253	.276	.294	.307	.318
C	.033	.065	.097	.128	.158	.187	.215	.241	.266	.289

Rt.
.936
.936
.936
.936
91.2

Throughput at the hw level of the three user classes for a number of software servers going from 1 to 20.

Comparing the graph above, it is possible to understand how the software bottleneck due to a lack of “parallelism” in the application server software became with the increase of the splitting factor (from 1 to 20) a typical hardware bottleneck.

We hope that the above examples had given a flavor of the possible results of the markovian “product form” queue models applied to the “rightsizing projects”.

The performance specifications

Besides the performance service levels identified through tables and requirements, the participants had to present a Performance Assurance Plan detailed with milestones, related to payments and analytical models delivery under penalty (if in delay).

The SPE (Software Performance Engineering) approach has been set out together with an analytical one based on the "product form" and the markovian queue theory [8]. The specifications of the Performance Assurance Plan have been done in order to assure the use of the approaches and techniques seen in the previous section in the rightsizing project.

Other issues important for the rightsizing process

The following issued proved to be important in the specification of the rightsizing process:

- A good assessment of all the application software in order to calculate in FP the complexity;
- Extensive knowledge of all the commercial tools available for reverse and forward engineering of the software.
- Need to minimize the organizational impacts.
- Careful technical-economical evaluation of the times and cost to calibrate the bid.

The results

The results of the tender

The winner offer has proposed a degree of innovation and system integration greater then expected with deadlines shorter then the forecast, also thank you to the extensive use of workflow techniques.

The levels of productivity offered have been higher then expected, with possible relevant savings in the software development activities in the mid and long term.

The new architecture will allow increasing the efficiency and quality of the services: it will be possible to query the workflow database, containing the work status of all the service processes provided to the clients.

This new architecture will also allow an easy externalization of the data and applications (extranet) for the development of new services directly connected to Internet and to the users.

The technological environment proposed

The technological environment proposed is characterized by an extensive use of the workflow technology that will increase the efficiency. It has several application server running Windows NT and UNIX as operating systems.

It foresees in less than 3 year the elimination of the mainframe and the full migration of all the applications in the LAN environment using Oracle as DBMS and Java as the target language.

The actual state of the project

Actually the start up period is going to finish and within one year the new downsized applications will start to be released.

Lesson learned

The experience done in this case up to now, has been mainly on the preparation phases of the tender and on the start up period of the contract.
Anyway from this experience and others done in similar projects we can learn certain lessons likely to be used by other organisations or companies in the future.

In general we must consider that a classical outsourcing approach (not to do a rightsizing) allows savings up to 20% thanks to the "economy" produced by the sharing of expensive hardware and software resources already owned (typically mainframes). But that saving hides often an hidden cost: the most of the times the IT environment is not updated over time making the Client more and more less competitive or able to provide flexible services (for example with an extranet).

This type of outsourcing is usually effective for "stable and repetitive activities" and not at all for a rightsizing process that, instead, typically changes the system in an irreversible way and is full of "one shot" activities.
A rightsizing increases cost in the short period but allows saving up to the 50% in the long term. The rightsizing, moreover, make the Client competitive and able to provide flexible services; it uses strategically the Information Technology.

We have seen in the several projects done that only a new outsourcing approach can be effective for rightsizing processes allowing saving in the short and long term bringing all the benefits of the outsourcing and the rightsizing process above described.

This new outsourcing approach, in order to be successful, must be based on the "economy" produced especially by the sharing of valuable know-how and professional experience in similar rightsizing project owned by the Company and his specialists. Then it becomes essential to select the Company respect the above features.

References

[1] E.D.Lazowska, J.Zahorjan,G.Scott Graham, K.C.Sevcik:. *Quantitative System Performance* , Prentice Hall 1984

[2] G.Iazeolla, F.Marinuzzi, Ph.D.: *LISPACK*: *A Methodology and Tool for the Performance Analysis*, IEEE Transactions on Software Engineering,Vol.19, Num.5, PP 486-502

[3] F.Marinuzzi, Ph.D.: *Il Ridimensionamento dei Sistemi Informatici verso i Sistemi Utente-Servente*, II°Università degli studi di Roma Ph.D.Thesis 1992

[4] F.Marinuzzi, Ph.D.: *Il Rightsizing verso i Sistemi Client-Server*, CMG Italia Ottobre1994

[5] F.Marinuzzi Ph.D., S.Soliani: *A New Symbolic Package for the Definition, Analysis and Resolution of Markovian Processes: Symbolic and inductive Techniques*, ACM ISSAC 92 Berkeley

[6] C.Smith: *Performance Engineering of Software Systems*, Addison Wesley 1990.

[7] G.Franks, A.Hubbard, S.Majumdar, J.Neilson, D.Petriu, J.Rolìa, M.Woodside***:*** *A Toolset for Performance Engineering and Software Design of Client-Server Systems*, Systems and Computer Engineering 1994 Carleton University.

[8] J.P.Buzen, *Computational algorithms for closed queueing networks with exponential servers,*Commun.ACM. , 16(9):527-531, Sept.1973

Appendix - 9

The Curriculum of the Author

The author has more then 10 years of experience of consulting in information technology and strategical planning at international level. For two years he has been the James Martin responsible for Italy.

He is the consultancy director of C.I.M. an active consultancy Italian firm. He has authored many papers for national and international conferences and magazines receiving prizes (for example.. IEEE TSE, ACM ISSAC Berkeley, AICA, CMG). He has a master in computer science and a Ph.D. with a thesis on Rightsizing versus Client Server system. For further information see http://www.marinuzzi.com [3].

The Company C.I.M.

The company provides outstanding consultancy services in the following fields: software engineering, capacity planning, feasibility studies, organizational and strategical consultancy all over Europe. In the past years it has provided training for several big Italian and multinational companies. It cooperates with market leader organizations such as Iter, Systech, Duke, (www.duke.it, www.iter.it). C.I.M., has been involved in several projects for leading private and public organizations (for example TIM, Procter and Gamble, TELESOFT, Public Institutes and Organizations, etc). Its mission is “simplify the complexity of the transitions processes due to technological innovation or organizational change”; it also supports the Clients in the planning and monitoring of the change projects or tenders.

It is focused in the field of the organizational restructuring, rightsizing and generally the migration of “legacy systems” to new architectures with special attention to best software engineering methodologies, approaches and tools. Internally it is a network organization with a database of more of 400 independent specialists. For each specific project, the best consultants are chosen from the database depending upon their past experience, specialization on the topic and independence.

C.I.M. believes firmly that its high competence and full independence from any vendor are the basis to build with the Client the trust necessary to accomplish the most difficult roles entrusted.

Session 10
SPI and Measurement I

Chairman
Risto Nevalainen
STTF, Finland

Applying Gilb's method of inspections into telecommunications software development

Dimitrios Stasinos
INTRACOM, S.A Peania, Attica, Greece
Georgios Tsoubelis
INTRACOM, S.A Peania, Attica, Greece
Vasilios Zioutopoulos
INTRACOM, S.A Peania, Attica, Greece

Abstract

This paper describes experiences acquired from a Process Improvement Experiment funded under the European Systems and Software Initiative (ESSI). The experiment, called GINSENG (Gilb's Inspections for Software Engineering), introduced Tom Gilb's method of inspections at Intracom's Software Development Centre. Through this inspections framework, Intracom aims to improve its current practices for telecommunications and other embedded software development by increasing the effectiveness of early defect detection and prevention activities. Additionally, suitable inspections measurements support improvements in the inspection and software development processes.

Introduction

This report describes acquired experiences during the execution of the ESSI Process Improvement Experiment GINSENG (Gilb’s Inspections for Software Engineering). The objective of this ESSI experiment is to establish at Intracom's Software Design Centre (SWDC) a systematic framework for software inspections based on Gilb's Inspection Method.

The initial step of GINSENG consisted of introducing the experiment within a digital telephone switch baseline project implementing new functionality in an incremental way. Thus, training in Gilb's Inspection Method was carried out and the inspection procedures for the baseline project were documented, by adapting Gilb’s method to the standard development procedures used which ran in parallel with the baseline projects’ implementation of early increments. Inspections were then executed throughout the baseline project’s latter stages (increments). PIE results were evaluated based on appropriate measurements. Finally, the appropriateness of Gilb’s inspection method for Intracom’s software development environment (mostly developing embedded systems) was evaluated, contributing to a series of internal and external dissemination activities.

Gilb’s Inspection broad interest and wide applicability support the transferability of the GINSENG experiences. Internal dissemination actions address Intracom Group of companies, while external ones Greek SMEs and Intracom’s European business partners.

Background Information

Objectives

The objectives of the GINSENG PIE can be grouped as follows:

Establishment and evaluation of a framework for performing software inspections based on Tom Gilb's inspection method, covering all development phases and work products

Integration of Gilb’s method with the software development processes currently in use, addressing technical, administrative and people issues

Increased effectiveness of early defect detection, reducing this way dependency on testing for product verification and validation

Gradual shifting in emphasis from failure recovery towards prevention of defects

Improved reuse and exchange of technical experience

Increased development process stability, improving control and facilitating further improvements.

Similarly the main commercial objectives behind the experiment are the following:

Cost reduction of software development

Increased productivity in software development activities

Reduction of time-to-market through less rework and less testing

Improved reliability of software products when these are in customer use

Dissemination of experience related to the application of Gilb's inspection method towards other software producing companies in Greece and internationally

Starting scenario

Intracom S.A. is the leading Greek telecommunications and electronics industry. Intracom's main software development activity concerns the development of high quality, digital telephony software products running on Ericsson's AXE-10 telephony exchanges. The SWDC, Intracom's software development centre, employs over 200 highly qualified and specially trained software engineers. Newly recruited software engineers undergo an intensive initial 3-month training course on the development method, the tools, and the application field. Additional training is provided based on project needs, covering major technological developments and evolving processes in relation to the application area.

Inspections are carried out using a process integrated with the software development model in use. These inspection activities are based on Fagan's model and have provided positive indications, in detecting and removing defects from software products. In other areas of the company/corporate group, where software development is carried out, the use of inspections is limited. Based on current experience, the usefulness and efficiency of inspections seem to vary rather significantly. Acceptance of inspections by involved personnel, as well as personnel motivation regarding inspections, differs significantly from project to project. Training on inspection methods is rather elementary, especially with respect to adapting inspections for specific needs and optimizing their performance. Particular weakness is identified in considering human aspects, team co-operation and management aspects. Efforts are under way to provide better feedback about problems and attitudes on conducting inspections, improve planning of inspections, balance available resources and ensure adequate inspections preparation time.

An independent quality assurance function has been established in SWDC, supporting software development projects, co-ordinating quality system documentation, performing audits and supporting corrective actions, as well as carrying out measurements of quality. There is a basic set of metrics that are being used providing a high-level view of the design process : lead times, effort/cost, as well as product quality/fault density. These measurements, taken from the literature, are providing useful overview and a basis for benchmarking. In addition, a more thorough metrics approach has been introduced to support improvements, based on GQM (Goal-Question-Metric) and *ami* methodology (through the ESSI PIE "PITA").

Intracom's practices for software development and project management have been analyzed, using as a basis requirements in ISO 9001/9000-3 and models for software process maturity (such as Capability Maturity Model - CMM).

Work-plan

The GINSENG work-plan included an initial set-up stage, 4 main phases (Introduction, Process Adaptation, Execution and Evaluation), and a final conclusion stage (more details in section 4.3).

Expected Outcomes

From a *software engineering point of view*, expected benefits from introducing Gilb's inspections include:
Improved quality in products
Facilitating prevention of defects in requirements analysis and subsequent development phases
Identification and triggering of process improvements
Achieving goals and controlling processes
Personnel training and motivation
Spreading of experiences to other software development areas of Intracom
Technology transfer related to improved inspections

From a *commercial point of view*, the impact of the PIE has several aspects, related to improvements in the quality of products and processes, and especially in increased productivity. By the term "quality of process", we mean characteristics s.a. repeatability, effectiveness and efficiency, controllability etc. that pertain to the development process itself and its stakeholders (mainly managers and developers), which of course indirectly, impact the product quality as well which is experienced by the end-customer. More specifically, expected benefits from the implementation of the proposed PIE include:
Reduction of overall software development cost
Cost reduction of rework, testing and initial maintenance activities
Shortening of the time-to-market for software products
Increased productivity in software development
Increased software product reliability when the system is in customer use

Work Performed

Organization

GINSENG was performed through co-operation of 3 different Intracom departments:
The Development Programmes Department, providing the overall managerial function as well as the liaison with the CEU.
The SWDC which provided the baseline project activity, had some of its software engineers receiving the Gillb's inspections training and was the recipient of direct internal dissemination of GINSENG experiences.
The Quality Assurance Department, with its Software Quality Assurance section, which provided technical support to SWDC and acted as a facilitator for introducing GINSENG smoothly in the context of the baseline project.

Technical environment

No unusual provisions were made affecting either the everyday work in the technical environment of the baseline project or the line management operation involved in the experiment. There is a database available to support collection, analysis and reporting of measurement data. Beside that, simple forms are provided to members of the baseline project which enable data analysis.

An infrastructure for publicity and internal dissemination has been implemented as an Intranet (WWW-like) environment. Moreover all documentation, relevant to Gilb's method, and tailored to the needs of the baseline project are available through the web, to both baseline projects participants as to the rest software design project members.

Phases of the experiment

In the context of the GINSENG experiment, Gilb's inspection method has been appropriately integrated with the selected baseline project. This integration involves three stages: 'initial', 'main' and 'concluding' stage.

The **initial stage** involved the establishment of a solid foundation for carrying out the experiment. More specifically, it consisted of training of the involved personnel, establishment of an inspections facilitator role (change agent) in Intracom's SWDC and handling of co-operation issues with the subcontractor(s). The purpose of training in this stage was to introduce and promote the inspections philosophy to senior, middle and project managers.

The **main stage** of the GINSENG PIE, corresponds to the implementation of Gilb's inspection method and involves four phases:

Phase 1. *Introduction.* This involved intensive training in Gilb's inspection method before the baseline project started. This training addressed several roles involved with inspections in the baseline project, including authors, inspectors/checkers, inspection leaders, project leader and quality assurance personnel. Additionally, training was provided to other key personnel, in order to facilitate internal dissemination and future application of Gilb's method to other projects.

Phase 2. *Inspections process definition.* During this phase, the inspection procedures for the baseline project were documented. Documentation here includes inspection process descriptions, role definitions, document check-lists and several types of forms. These documents were reviewed before being issued by both project participants as well as other key personnel that have attended training in Gilb's method. During the documentation of the inspection process, aspects of the standard development method were taken into account. Additionally, appropriate inspection measurements such as faults per page, checking rate, estimated remaining defects after an inspection and defects per page found during test or field use, were defined (based on the *ami* method). An inspections database was set-up for subsequent storage and analysis of inspection measurement data. Finally, other measurments were adopted in order to help the implementation and monitor the achievement of technical and business goals (discussed in the Results and Analysis chapter).

Phase 3. *Inspection process execution.* In this phase, the inspection process defined in phase 2, is being used throughout the baseline project. Each time the baseline project reaches a milestone, a review of GINSENG progress is also performed. Measurements are collected as planned in phase 2 and measurement results are included in the relevant measurement reports as these become available.

Phase 4. *Inspection process evaluation*. This phase corresponds to testing and initial maintenance activities in the baseline project. During these activities, no more inspections are performed, but defects not found by inspections are detected and removed. Data collected during the period the system is tested or is in customer use enable a thorough analysis concerning product quality and inspections effectiveness.

The **concluding stage** of the GINSENG PIE includes an evaluation of the effectiveness and efficiency of Gilb's method. To this purpose, the measurements taken from the baseline project will be analysed both alone and in association with similar data from past projects (not using Gilb's method). Based on evaluation results, the inspection process definition documentation (from phase 2) will be appropriately updated.

The baseline project started at project month 6 (when Inspection Process Definition is well under way) and ends in project month 17 (with a shift of up to 10 weeks due to a prolongation of the baseline project independent from the PIE), to be followed by the experiment's 'concluding stage'. Inspection Process Execution and Inspection Process Evaluation will take place in parallel to the baseline project.

Results and Analysis

In relation to objectives set forth for the experiment, measurements (derived by the GQM/*ami* method) available to date are discussed below:

Regarding the integration of Gilb's inspection method with the existing development process there are 2 metrics:
the *number of problem reports* caused by GINSENG implementation; in this case no such report has been produced to date. This is considered as a result of the proper and detailed Gilb adaptation performed for the standard development and (to a small degree) for the baseline project.
the *overhead caused to standard activities* (percentage of additional effort). This overhead is estimated currently at to be just under 8% of the scheduled project effort, which is well within the 10% expected upper limit for the first (pilot) implementation. A reduction trend with each future implementation is expected.
There is not yet objective value of the *ratio of 'major defects" in inspections over the total detected defects* (in inspections, testing and 3 months in customer use, related to objective 3 of the PIE). Measurements are now being collected in the baseline project and there are no reliable results yet. As an indirect interim measure, we use the fault density of inspections-detected effects currently being collected, which shows a significant 147% increase (to a level of 2.37F/KLOC). This is an indication of significant improvement over the current level of 26% for the aforementioned ratio (but has to be verified with complete test and field use data).
There is also no accurate and stable value that can be reported yet for *person-hours of development rework per project*. There is need to include data from correcting errors found during testing (being carried out) and field use and ensuing rework, but on the other hand there was an increase in the density of defects detected(not started

yet). Some preliminary results indicate a decrease in such late defects during design and implementation by inspections (147% more than usual), thus there is no clear trend at the moment. See also charts included later on in section 5.1.

In section 5.1 below there is also a chart for *Rework time probably saved=(#defects*20)-(Total Inspection time) in mhrs per KLOC,* averaged for all inspection items. Another chart is for *Rework saving ratio=(rework time probably saved)/(total inspection time),* again averaged for all inspection items. The actual return-on-investment is even higher, if one accounts for the fact that field-detected defects cost far more to rework/correct, even not counting customer satisfaction aspects.

Measurements have started to be taken but no stable value can be reported yet for *fault density* (number of faults per 1000 lines of source code) *during the testing activities and during the first 3 months of system field operation* (related to objective 10). Preliminary indications show an improvement in test fault density (probably to below 1/KLOC from 1.3 historical average, but this is premature and no field data are available yet.

Another measurement to be used for overall evaluation at PIE conclusion is for lead-time the *number of days from baseline project initiation to delivery of products to the market*. This figure is dependent for each individual project on a number of external events that can occur (e.g. change of requirements, delays in customer review process etc.), leading usually to a delay in delivery. Thus there is significant variation. And trends can be followed in the long run. For the baseline project, there was no delay in normal scheduled time due to inspections. Still, a delay was caused by a reshuffling of requirements implemented across successive increments impacting also the duration of testing phases and delivery dates, quite independently from inspections. This will prolong the baseline project duration for up to 10 weeks (according to revised schedules) and will delay also similarly the collection of test and field data for the PIE and the concluding phase.

Further particular results are described below for the respective impact areas:

Technical

Important results which have been acquired (or are in the process of collection) from the GINSENG experiment include:

tailoring guidelines for adapting the standard software development processes.

forms used on-line for feeding-in detailed data required for each inspection.

A report from both training courses which provide all relevant aspects of both the method, further issues and applications of the method from what the baseline course is about and finally feedback from all participants.

Carrying out of Gilb inspections in a seemingly effective manner after some initial problems were resolved with the help of mentors.

A further result is expected to show up in future projects, i.e. quality gains and process improvement, taking advantage of results and analysis achieved by Gilb’s method in the baseline project. Such analysis is already carried out in a short brainstorming session after each inspection meeting in relation with the inspected item and defects detected thereof, while an integral analysis will be carried out with the baseline project conclusion. The analysis performed after each inspection provides a general categorisation of the type of each defect, as well as a root cause analysis for defects considered very significant and/or complex is encountered in order to propose an improvement for the future. A compilation of such analysis results will be provided at the project conclusion by the technical coordinator for the benefit of future projects. The following defect classes were defined:

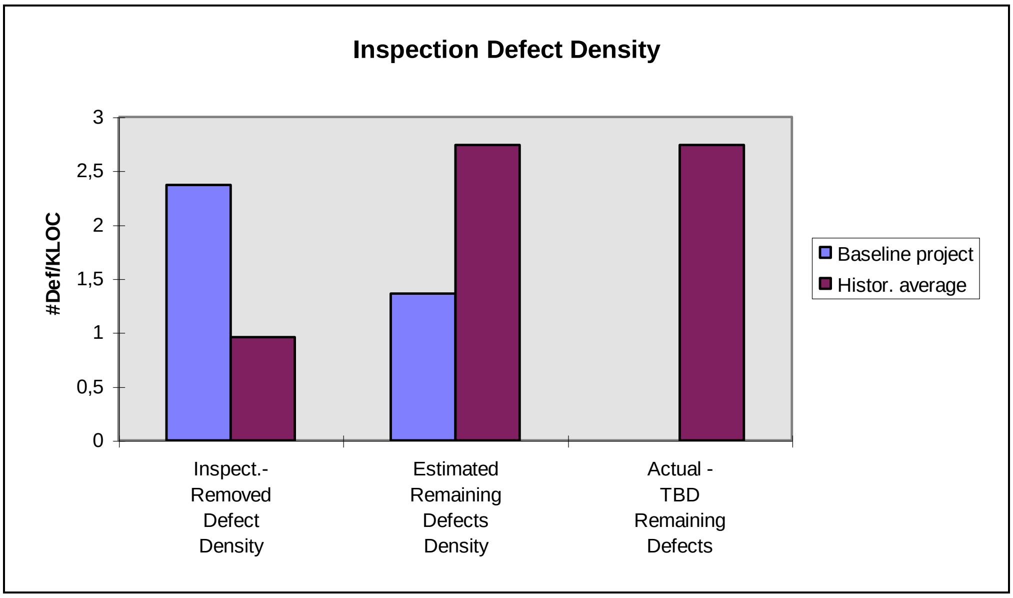
In “Technical” class: functional/algorithm, data, parameter, interface, signal, other (a few additional technical categories are also used for particular types of inspected documents, allowing for more focused improvements).

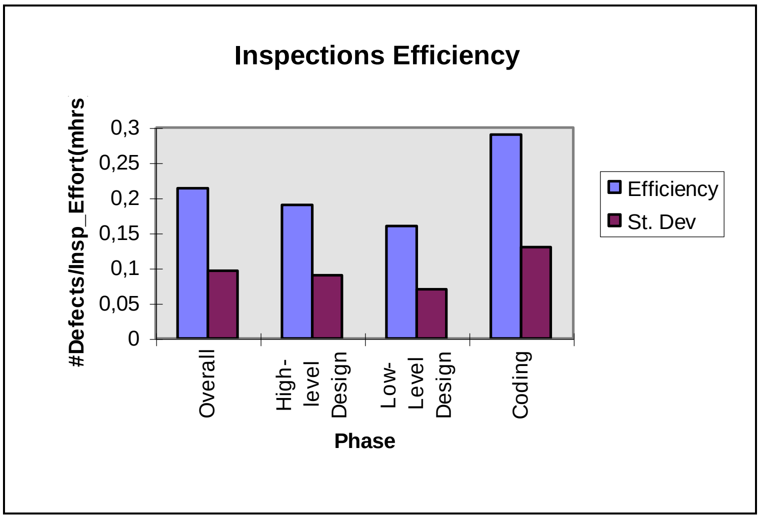
In “Administrative” class: inadequate experience/training, haste/omission/oversight, inadequate/erroneous spec (in previous design phase), inadequate/erroneous_advice/help.

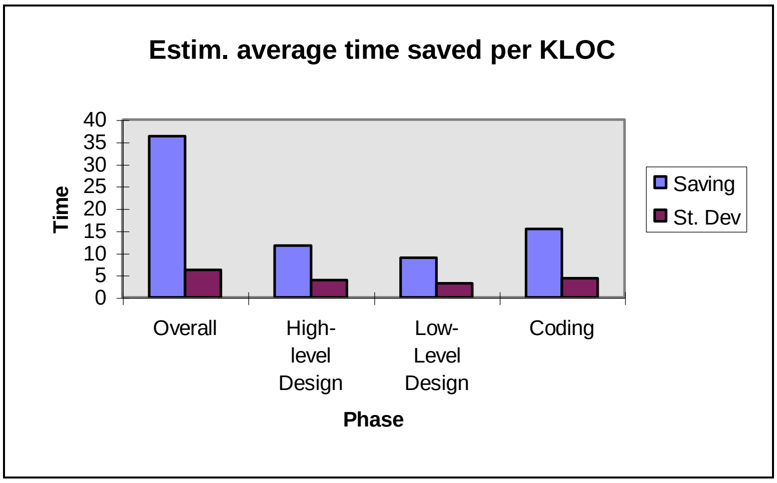
“minor” (not causing failures) class: such defects are only a by-product of the inspections process (their detection is just registered and not emphasized), are not included in overall statistics since they are not part of the process under control (feedback from test and field failures), but are measured separately, as a secondary/idirect indicator of quality

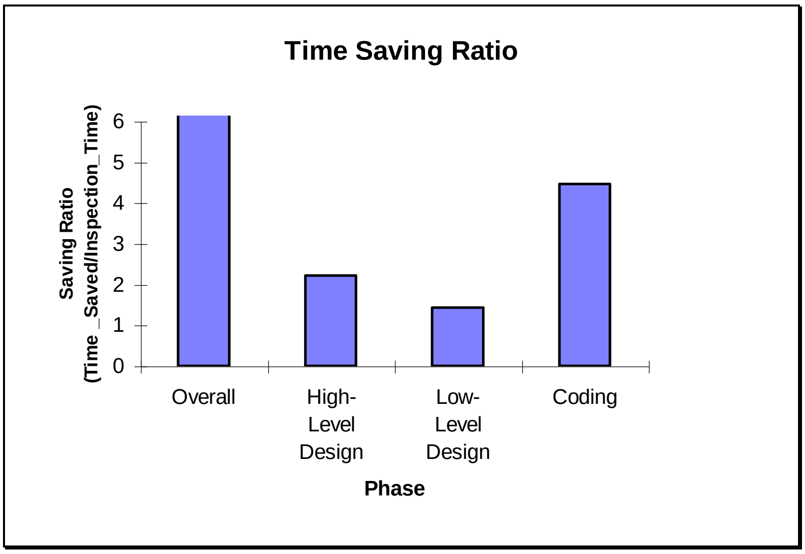
Such categorization will provide in the end an overall picture in relation to maturity of planning the design process, technical maturity of human resources and of planning overall. These are to be compiled only at the general project level (not at individual designer level) to avoid any misunderstandings and suspicions.

Measurement results are being obtained and analysed, from the baseline project. Some consolidated charts reflecting measurments as described before, were prepared by both mentors and the technical coordinator are included below.




Inspections Efficiency
#Defects/Insp_Effort(mhrs
0,3
0,25
0,2
0,15
0,1
0,05
0
Overall
High-level Design
Low-Level Design
Coding
Phase
Efficiency
St. Dev


Estim. average time saved per KLOC
Time
40
35
30
25
20
15
10
5
0
Overall
High-level Design
Low-Level Design
Coding
Phase
Saving
St. Dev


Time Saving Ratio
Saving Ratio
(Time _Saved/Inspection_Time)
6
5
4
3
2
1
0
Overall
High-Level Design
Low-Level Design
Coding
Phase

measurement results are collected from other ongoing improvement activities related to the application of the method of Gilb (like policy deployment actions in relation to review effectiveness). At the conclusion of the PIE these will help evaluate the overall improvement capability of Gilb's inspections.

Business

The impact of the GINSENG experiment to Intracom's business operation can be considered as indirect but potentially significant. It is very important, for business operation, to be able to base the improvement efforts on the individual 'front line practitioners'. This will improve the product quality of the products that the customer will experience, benefit from and appreciate, thus enhancing the competitive position of the company.

We anticipate gains in productivity (measured in LOC output per effort expended overall during development cycle (including testing) and time-to-market. The main factor contributing to such improvements is expected to be less rework (due to less defects slipping through to later phases where remediation becomes far more expensive).

A final aspect concerns the enhanced capability, team enforcement and consciousness of the personnel, gained through GINSENG, which can offer another significant competitive advantage for Intracom in a business where human capabilities and knowledge are crucial for success.

Organisation

The organisational changes effected to support GINSENG activities are summarised in the following points:

A GINSENG team was formed, including: baseline project personnel, two facilitators (acting as mentors), and a technical co-ordinator that needs to have an overall view of the work in progress.

Monitoring the Gilb's Inspections method introduction is a responsibility of the SEPG already established in the SWDC. This is in line with SEPG's role as promoter and co-ordinator of the SPI efforts, in this case focusing on the individual rather than other organisational entities. In this respect, it is of particular importance that the Gilb's Inspections experience augments and complements efforts already undertaken for SPI. The method of performing inspections via the method of Gilb is fully compatible with CMM based improvements already under way in the SWDC, as it has a direct involvement to peer reviews (CMM level 3) and defect prevention (CMM level 5).

Culture

There is only limited experience to date (only selective/limited application of APPLYING Gilb's inspections) for Intracom to assess the cultural impact of the

GINSENG experiment to the software development personnel (managers and engineers). Even so, some elements of a cultural shift have been introduced and favourably accepted by GINSENG participants, while even more significant potential exists as indicated below:

Gilb's inspections provides a much needed framework for teamwork in line with business/unit goals and objectives
Team efficiency and improvement aspects are significantly promoted. As already mentioned, execution of inspection with Gilb's method leads to systematic team process improvement by providing a robust model of team operation and effectiveness.
A basic reference on SPI is provided for everyone to use and an effective language for systematic improvement becomes (potentially) everybody's knowledge in their standard every day work.
Intracom expects the method of Gilb to be a strong foundation for supporting, in a bottom-up manner, a universal continuous improvement culture, by providing specific activities that implement top level goals related to quality and productivity.

Skills

During the initial phases of the GINSENG experiment, significant effort was expended in training members of the baseline project, as well as key personnel across the SWDC. The training covered techniques and disciplines useful in implementing inspections via Gilb's method. Training involved 20 people to date, in two sites of INTRACOM where the baseline project is executed, while more people will be trained in the future.

In addition, line and project managers were offered a overview training (also delivered by the Ginseng's technical co-ordinator) exposing them to the inspections via Gilb's method basics. A significant number of managers are trained to date and more will be trained in the future. These people will be trained to understand the benefits of performing inspections via Gilb's method and to be supportive of individuals and teams applying this method.

Finally, the baseline project and other involved personnel has acquired expertise in introducing Gilb's method and its consequent metrics. This expertise is disseminated to other project teams within Intracom.

Key Lessons

Technological point of view

In general, the Method of Gilb shows good value potential for beneficial application to software development.
At the training stage, the courses provided, initially to Ginseng technical co-ordinator and then two the two mentors was found to be quite educational as well as definitely demanding and challenging for every participant.
In relation to actual implementation of the Gilb's method, after process adaptation and guidance was provided, no serious problems were encountered to date.
In general, it appears that Gilb's method is more easy to introduce and may

demonstrate higher ROI (Return On Investment) in the short term, in software development areas where short and recurring development cycles are involved, rather than in environments with long software development cycles. This is due mainly to the fact that in this (short cycle) case, adjusting and learning of the method should produce results sooner, e.g. from the second application cycle after the pilot introduction, while thereafter, results from defect prevention activities (analyses, guidelines etc.) will accumulate in a much faster pace and be based on a wider spectrum of development cycle experiences. Intracom intends to introduce Gilb's inspections to short cycle development, besides the baseline project experience.
On the other hand such an intensive, thorough and systematic approach to inspections with particular emphasis in defect prevention can be very powerful when very high quality standards apply (e.g. military, aerospace, telecom, medical, etc.), normally associated with long development cycles. This indeed is a strong motivation to adopt such a method in cases like Intracom's baseline project as well as other similar areas (telecom, safety, military or other real-time software), where high reliability is pursued (even with a higher initial cost). Indeed, ROI even in this case could rise significantly if one counts the negative business impacts of delivering less than top-quality software to the customer.

Business point of view

Justification of introduction of Gilb's inspections. The initial effort expended for Gilb's inspections establishment, represented a rather medium overhead. The overhead from introducing Gilb's inspections was justifiable only in a wider organisational context, based on the prospect of introducing Gilb's inspections gradually and eventually for every new SWDC project as a standardised suggested process. For the foreseeable future however, this has to be applied on a volunteer basis (at project level). The issue of introducing Gilb's inspections at the SWDC was under serious consideration prior to GINSENG, but GINSENG itself accelerated the implementation of Gilb's inspections.

Spreading the use of Gilb's inspections. The Gilb's inspections experiences acquired through GINSENG, appear to be spreading and considered for reuse and adaptation in other software development areas as well (new projects both within and outside AXE-10 development). It is foreseeable that Gilb's inspections may become institutionalised first in certain application areas (first of all in the one that the baseline project belongs) then in others and eventually overall in the SWDC. Acceptance and support of Gilb's inspections has to be ensured at each stage before proceeding further towards an institutionalisation path. Expanded use and acceptance of Gilb's inspections enhances the impact on the original application area which will be viewed as a pioneering effort.

Project overhead. The project overhead caused by the first implementation of Gilb's inspections, based on baseline project data, was estimated to be under 8% excluding initial training. It is expected that this overhead will drop significantly in future cycles where Gilb's inspections are to be implemented. Actually, it is expected that when the method of performing inspections according to Gilb gets institutionalised (used as a standard inspection process upon the majority of the projects) will remain low, at a level of around 3 to 4% or even less, if full automation in data collection and analysis can be provided. This appears to be an acceptable cost with high gain in quality over the whole software development cycle.

Outside interest. This has been expressed for Intracom's GINSENG related experiences and there is potential for Intracom to provide a pioneer method for inspections and SPI related services to the Greek market of software developers. GINSENG related experiences and practices should be carefully and gradually transferred in the environment of different organisations. This is because of potential differences in technical and business characteristics, and particularly due to a software development process which may be less mature than Intracom's one.

The method of Gilb complements other SPI activities. Being a generic and flexible approach, the method of performing inspections according to Gilb was found to be well suited for such a purpose, addressing the bottom line, i.e. the individual software engineer and software development teams. It is for instance, closely linked and complementary to CMM based SPI actions (SWDC is already actively involved in this).

Strengths and weaknesses of the experiment

A number of positive comments can be made on the approach followed by GINSENG, based on results and experiences up to project midterm. These can be summarised as follows:

Introduction of the Gilb's method in performing inspections was introduced to enhance the effectiveness and efficiency of the existing design review/inspection process. To this end the existing experience in inspections has been used as a comparison basis to debate the validity of the Gilb method, but on the other hand it has been used as a bridge for the better introduction and exploitation of the method, adopting a stepwise approach.

There is a positive perception by people involved in the experiment that the method of Gilb can help to improve their efficiency and team consciousness relative to inspections, also gradually building confidence that the possible faults slipping through subsequent phases are being reduced.

Introduction of defect analysis with potential of improving the software design process itself to prevent occurrence of faults in the first place with benefits to be achieved in future projects.

The nature of the method has made its tailoring to the existing processes of the organisation relatively straightforward, increasing somewhat the early appraisal and prevention time but with good prospect of significantly reduced late appraisal as well as failure/rework time.

Gilb's method complements other SPI initiatives and programmes already in place such as CMM (peer reviews at Level-3, and defect prevention at CMM Level-5), Competence Development, Policy Deployment etc., the approach taken providing mutual benefits with such efforts.

The method of Gilb has high potential to quickly spread to different software development areas in the company since the inspections' outcome ensures more reliable criteria for approval and acceptance of software design products and promises enhanced quality of such products.

The method of Gilb appears to have high potential for effective dissemination and consulting to other organisations that will be interested.

Adequate and effective management support was achieved and continues to help perform the experiment.

The method can be adapted for use in other areas of design work besides software; for example hardware when high and exacting standards are prevalent (this is a very important prospect, since Intracom is involved in several such areas).

On the other hand, a number of problems and/or limitations were identified in carrying out the approach followed. These can be summarised as follows:

Introduction of Gilb's method to a baseline project has to be performed in a 'discretionary' way, avoiding any potential disruptions. Introducing Gilb's method can be a risky undertaking for a project which is planned and initiated independently and without a pre-existing infrastructure and culture. In the case of GINSENG, for instance, there was no possibility to establish a baseline project dedicated to the GINSENG experiment. Thus, one of the normally scheduled and planned projects was used. This will be less of a problem in newer projects where adoption of Gilb's method will be achieved in a more educated manner and planning can be more effective, based on GINSENG experiences.

Some hesitation was expressed originally from a few practitioners (software designers) fearing a bureaucratic overhead on top of their technical work. This was gradually overcome with the help of mentors and as the practitioners themselves started to recognise benefits in the method from results to date and the empowerment that it offers them to perform their work in a better way.

References

[1] Fagan M. E. *Design and Code Inspections to Reduce Errors in Program Development*, IBM Systems Journal, 15(3), 1976, pp. 182-211

[2] Fagan M. E. *Inspecting Software Design and Code*, Datamation, 1977, p. 133-144

[3] Fagan M. E. *Advances in Software Inspections,* IEEE Transactions in Software Engineering, SE-12(7), 1986, pp. 744-751.

[4] Gilb, T., Graham, D., *Software Inspection*, Addison-Wesley, Reading, U.K, 1993.

[5] Gilb, T. *Advanced Defect Prevention Using Inspection, Testing, and Field Data as a Base*, American Programmer, 1991, pp. 38-45

[6] Gilb, T., *Document Quality Control: The case for quality control in written products and processes*. Computer Task Group's PEOPLEWARE, U.S.A, 1992, pp. 12-15

[7] Grady, Rober B, and Slack, Tom Van, Hewlett-Packard, *Key Lessons in Achieving Widespread Inspection Use*, IEEE Software, July 1994 pp.56-58

[8] Humphrey, W.S., *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1995.

[9] Marc C. Paulk et al. *The Capability Maturity Model, Guidelines for Improving the Software Process*, Addison-Wesley, Reading, Mass., 1995

[10] URL:www.ourworld.compuserve.com/homepages/kaiGilb.

Appendices

The Companies

INTRACOM S.A:

Founded in 1977, INTRACOM is the largest manufacturer of telecommunication equipment and information systems in Greece. In 1990, the company is listed on the Athens Stock Exchange and, by accelerating growth, establishes a strategic position within the European market. In cooperation with its subsidiaries and affiliates, the company provides products and services to the Greek public and private sectors, while developing significant international presence. INTRACOM provides products as well as integrated services for the design, manufacturing, turn-key project implementation and support in the following areas:

Public Telecommunication Networks - Telecommunication Systems Software - Integrated Business Networks - Network Management Systems - Energy Management Systems - Satellite Applications - Defense Systems - Integrated Wagering Networks

Authors

Mr. Dimitris Stasinos, M.Sc, is a telecommunications S/W development engineer in INTRACOM. Mr Stasinos has an experience of 3 years in development of real-time digital telephony applications, related to the area of ISUP (ISDN User Part). He has has also acted as Project Manager in process improvement projects. Today he holds a position as process responsible. Mr Stasinos was trained as an inspection expert by Tom Gilb and acts as a facilitator for the introduction of Gilb's inspections in INTRACOM. He can be contacted at dsta@intranet.gr.

Mr. George Tsoubelis, D.Eng., is a telecommunications S/W development engineer in INTRACOM S.A. He has a 3 years experience in the development of real-time digital telephony applications, related to the area of Operation and Maintenance and ISUP, while for the last 2 years he is acting as project manager in the ISUP area. Mr Tsoubelis is the project manager of the baseline project. He can be contacted at gtso@intranet.gr.

Mr. Vasilios Zioutopoulos, B.Sc., is a software quality assurance engineer in INTRACOM S.A. He has 8 years of experience in quality assurance support of internal projects and in software process improvement initiatives. He holds an executive position as responsible for the organisation and management of ESSI (Ericsson System Software Initiative) deployment within INTRACOM. Before that he has worked for Cussons International Ltd., a multinational manufacturer of personal hygiene products, as project manager in production line design and installation, in U.K. Mr. Zioutopoulos is a software inspections methods expert. He acts as the overall technical coordinator of Ginseng project. He can be contacted at bziou@intranet.gr.

Control your projects by improved planning

Henry Meutstege
Centraal Beheer, Apeldoorn, The Netherlands
Willem Bos
Centraal Beheer, Apeldoorn, The Netherlands

Introduction

Characteristic for many software development projects is the delay in delivery, the budget overrun and the delivery of a product which is not according the customer expectations. A number of the software development projects of insurance company Centraal Beheer suffer the same problems. The central IT department wanted to deal with these problems and decided to start a process improvement experiment. This experiment is supported by the European Commission. The experiment is called Plan-IT and is known by project number 27784.

In this article we will describe the process improvement experiment, our experiences with the supporting tools from Quantitative Software Management (QSM) and the lessons we learned during the experiment.

Centraal Beheer

Centraal Beheer (CB) is a medium size (3300 employees) insurance company with operations in the Netherlands. Centraal Beheer is an insurance company acting as a direct writer for life as well as non-life insurance. The company is part of the ACHMEA group in the Netherlands and the EUREKO group in Europe. EUREKO is a large international banking and insurance company with over 33000 employees situated in 12 different European countries.

Centraal Beheer, employs 150 people in its central IT department. Since 1997, it has adopted the Capability Maturity Model (CMM) as its process improvement model. As a result of CMM-driven improvement initiatives, the department has reached Level 2 in many areas of its development process. However, a new assessment showed that project planning and tracking still needed attention. These area’s of CMM are the focus of the Process Improvement Experiment Plan-IT.

The central IT department consists of several parts :
Application Services; This department is responsible for maintenance of several large central IT systems.
Application Component Centre; This department develops or composes new systems based upon components.
Electronic Banking Applications; This department develops systems based upon web-

technology.
Support; Supports the prior mentioned departments

The Process Improvement Experiment

In origin the experiment was initiated because of problems to realise projects within planned effort and time. This problem was indicated by our internal customers, who became more and more aware that our software development projects should run better. Our customers pinpointed weak planning and tracking as the mayor causes of their dissatisfaction.

With the process improvement experiment we want to improve two Key Process Areas of CMM level 2:
Project planning
Project tracking and oversight

Because we started our project PLAN-IT in august 1998, we decided to perform another (smaller) assessment to measure the baseline maturity level at the start of this project. This showed information about the areas we had to improve for a higher level and information about the changes in our organisation since the latest assessment in June 1997. This assessment also showed a low maturity on Software Configuration Management. We decided to postpone actions on the key process area Software Configuration Management until after this experiment. Our main goal was delivering a product within planned time, effort and quality.

The main objective of the experiment:

Improving the quality of the planning and tracking process in order to control the costs, the time to delivery and the quality of our software development projects.

Demands and wishes

In order to improve the areas of project planning and tracking we needed (historic) baseline information at the start of a new project. This information is necessary to organise a project, to make a founded offer to the customer, to make a realistic planning and to manage the project capacity. We wanted to be able to manage the expectations of our customers.

To realise the main objective we defined several actions:
Improve the current standard guidelines and procedures for project planning focusing on the appropriate use of historic data from completed projects.
Establish a project monitoring office with the following main tasks:
Supporting project management with planning, tracking and reviewing the software development projects;
comparing results with documented estimates, commitments, and plans;
providing adequate insight into actual monthly progress of the different developments, so that management can take effective action when a software project's performance deviates significantly from its planning;
The project monitoring office (together with the project managers) will have the responsibility of quantifying and assessing the risks at each stage of development.
Experiment in a real life planning & development situation with the new approach and the support of an estimation- and tracking tool.
Create awareness of the importance of the use of effective planning and registration techniques.

The starting point of the experiment was to improve and test the procedures and guidelines and test the selected supporting tools in two real life software development projects. The first software development project was used to develop new guidelines and procedures and improve the existing ones. Also the supporting tools were tested during this software development project. During the second software development project the guidelines and procedures were fine-tuned and the tools were implemented in the organisation.

Project monitoring office

During the software development projects the Project Monitoring Office (PMO) had to be organised. The PMO played a very important role in the experiment. The activities of the PMO were clearly defined. At the start of a software development project, the PMO office delivers several services (like function point analysis, review of the project plans and create several project scenario’s using the QSM tools). An advantage of this approach is that the PMO is involved in the starting phase of a project. This way it is able to show the advantages to the project managers in an early stage. The project managers were very satisfied with the support by the project monitoring office.

The office supports the project in:
Organising the project
Organising quality in the project
Calculating the project
Making a planning
Tracking the project
Collecting metrics

In the past historical data wasn’t used in new projects. During the experiment we started collecting metrics in order to learn from the past and to estimate new projects, based on this information. Collecting this information is a major change in our organisation and in particular to the project managers. People had to get used to this new attitude of collecting metrics. The attitude slowly changed. Collecting and keeping these metrics is one of the tasks of the project monitoring office.

We also centralised the use of the QSM tools at the project monitoring office. The people working at this office are trained in using these tools. Training of other employees is not necessary. Unambiguous use of this tool is extorted this way.

Supporting tools

In order to support us in the planning and tracking process we needed an estimating tool and a tool that supported us in tracking a project. We wanted to make a founded offer towards our customers. We also wanted to spot possible extension of a project in an early stage, not at the moment it occurred. With these demands and wishes we examined the possibilities of several supporting tools. Only few seemed useful in supporting our goals. Among them Knowledge Plan of Software Productivity Research (SPR), Seer Metrics of Seer, and the SLIM tools of Quantitative Software Management (QSM). Examining these products resulted in the conclusion that only the Slim (Software LIfecycle Management) tools of QSM satisfied our needs. Knowledge Plan and SEER Metrics had no facilities to support the project tracking process.

Historical information of company-own projects can be imported in the QSM SLIM tools. This way the tools can calculate the Process Productivity of your own organisation. The Process Productivity is indexed within the SLIM tools by the Productivity Index (PI). The equation which calculates the process productivity was developed by the founder of

QSM, Lawrence H. Putnam. It is based on 4000 software development projects. It has the next formula:

$$\text{Process Productivity} = \frac{\text{Quantity of product (in source line of code)}}{(\text{Effort in person years/Skills factor})^{1/3} * (\text{development time in years})^{3/4}}$$

An description of the software equation and the philosophy behind the QSM SLIM tools is found in "*Executive briefing: Controlling software development*" by Lawrence H. Putnam.

The QSM SLIM tools

The QSM SLIM tools is a tool set with :

SLIM Estimate

This part of the SLIM tools generates Estimates of Software projects. One can produce several different scenarios of a software development project. The necessary input consists of:

Size of the system (in Lines of code, number of objects or function points)

Application type (business, telecom, production etc.)

Productivity Index (PI, number that reflects the productivity of the department)

The tool generates information on time (person months, milestones and phases), costs and quality.

SLIM Control

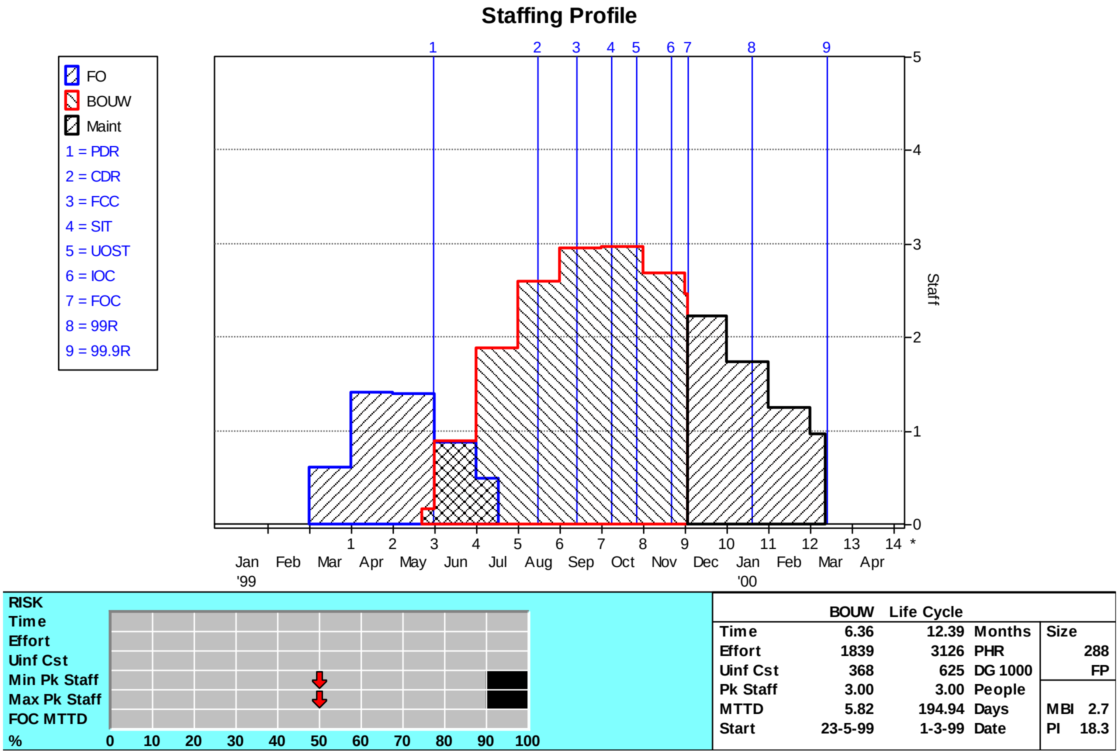
SLIM Control can be filled from SLIM Estimate with an initial plan. During the project, tracking data must be entered. Every period data like realised hours, realised progress in functionality and number of errors found during testing are entered. By entering this information, SLIM Control calculates the progress, compares the progress with the initial plan and is able to forecast the further development of the project. After only three or four measurements in the construction phase you can see how the project will develop in the future.

Slim Metrics

In this tool extra metrics are collected, analysed and transformed into useful (management) information. This tool isn't evaluated during the experiment.

SLIM Estimate

The tool Slim Estimate uses three parameters: the size of the application, type of the application and the Productivity Index (PI). The size of the application must be put into the tool in lines of code or function points. For the type of application you have for example Business, Process Control, Scientific and some other similar types. At Centraal Beheer practically all applications are of the Business type. The Productivity Index is calculated by using information of the QSM history base and the own history. When there are no further conditions Slim Estimate is capable in presenting a project scenario. An example of the output of Slim Estimate is presented in figure 1. This is only a small part of the total available information.



	BOUW	Life Cycle		
Time	6.36	12.39	Months	Size
Effort	1839	3126	PHR	288
Uinf Cst	368	625	DG 1000	FP
Pk Staff	3.00	3.00	People	
MTTD	5.82	194.94	Days	MBI 2.7
Start	23-5-99	1-3-99	Date	PI 18.3

Figure 1. Phaseplan from Slim Estimate

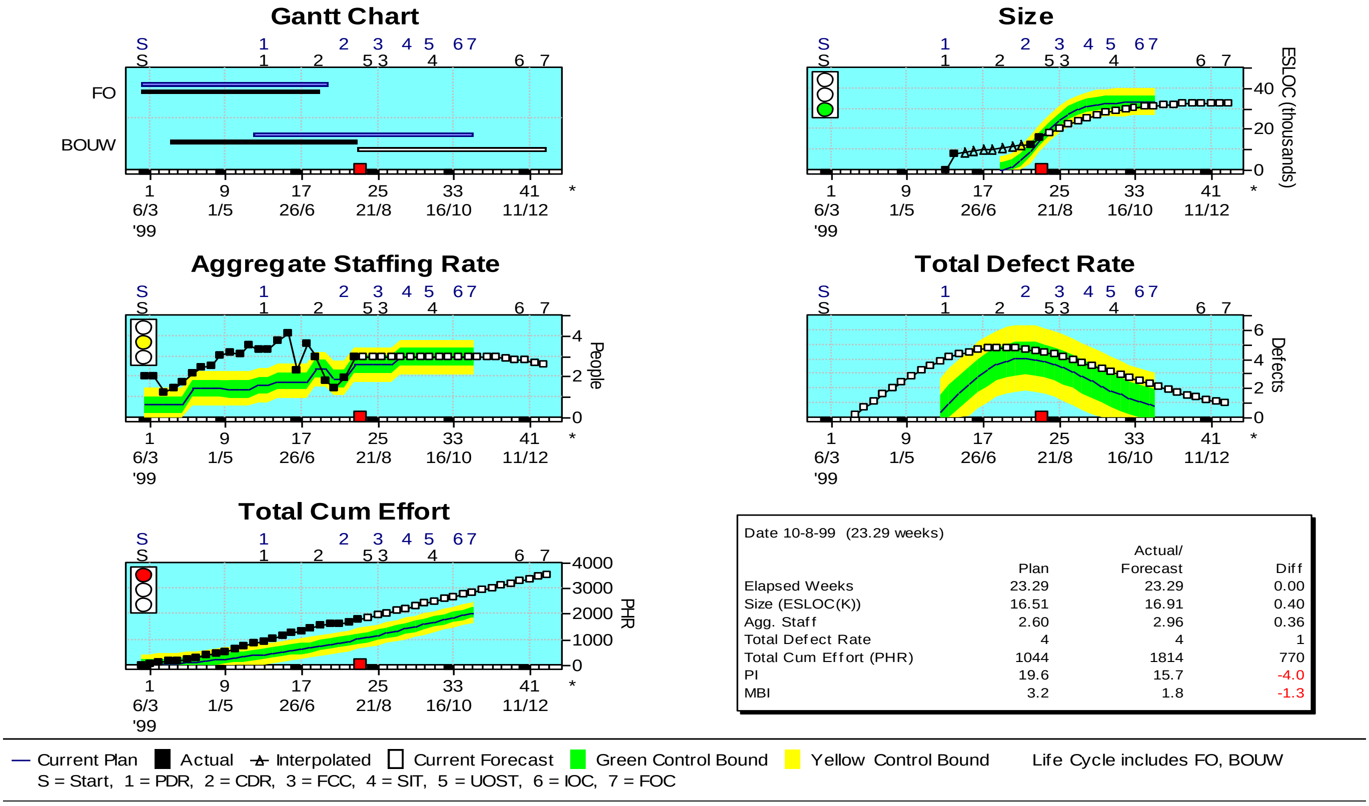
By playing with the basic elements (costs, time and quality) of a software development project different scenario's can be produced within little time. These scenario's can be used as input during negotiations with the customer. You are able to show the customer the consequences of his question(s). When the customer changes his demands, you can show the project changes in:

Time
Effort
Milestones
Phases
Risk

Because of this improvement the communication with our customers has improved. We are able to manage their expectations.

SLIM Control

The second tool SLIM Control uses the chosen scenario from SLIM Estimate as base. Every two weeks the progress has to be put into the tool. The progress in realised hours and realised functionality is measured. Realised functionality can be measured by counting realised lines of code, or realised function points. As a result SLIM Control compares the realised progress with the baseline plan of the project. An example of the output of SLIM Control is presented in figure 2.



Date 10-8-99 (23.29 weeks)

	Plan	Actual/ Forecast	Diff
Elapsed Weeks	23.29	23.29	0.00
Size (ESLOC(K))	16.51	16.91	0.40
Agg. Staff	2.60	2.96	0.36
Total Defect Rate	4	4	1
Total Cum Effort (PHR)	1044	1814	770
PI	19.6	15.7	-4.0
MBI	3.2	1.8	-1.3

Figure 2. SLIM Control output

SLIM Control warns by traffic lights when one of the basic elements is running out of plan. A green traffic light indicates a good realisation according to plan. A yellow traffic light indicates that this element needs attention but still is in the safe zone. And a red traffic light indicates that the realisation is not according to plan and needs immediate adapting.

Implementation of improvements

Create awareness of importance of effective planning and registration techniques
The management initiated this experiment because of problems in projects. The main problem was that the project manager didn't have control on the project. At the beginning people were sceptical towards the experiment but now they see the first results this attitude changes into a more enthusiastic attitude.

Disseminate results
We disseminate the results of the experiment as much as possible. We have had a lot of conversations with both management and project managers about how to organise a project and cope with planning, tracking and registration. We use different media in which we disseminate e.g.:
Achmea Newsletter
Kick off meeting
Self assessments
Software Process Improvement Newsletter (SPIN)
Measuring report '98
Conferences (Fesma, EuroSPI, SERC Software Process Metrics)
Article at Software Release Magazine
Software in Focus Newsletter

Reward suggestions
Another way to encourage people is a small present they can win when they have a good suggestion to improve the process. This way people start thinking about the process.

Project monitoring office
The project monitoring office has a central role within the implementation of improvements. This office can locate immature areas in the process. The supporting task of the office helps in implementing improvements. At the beginning the office actively helps the projects in planning and tracking. When project managers get used to the new approach they can do the work themselves.

TIP!
Involve people in the improvement experiment
Keep people informed
Reward suggestions for improvement

Experiences

During this experiment we had some useful experiences. These experiences concern involvement of the management, the project monitoring office and the QSM tools.

The project monitoring office had a central role in the experiment. It took care of registration of data and feed back to the project manager. This office was also responsible for the dissemination of the results and awareness within the organisation. The knowledge about the QSM tools was centralised in the project monitoring office.

The project monitoring office can generate management information out of the project information. This is important to get management commitment for an improvement experiment like Plan-IT. If this commitment fails, the experiment has no chance to survive. The management has to propagate clearly this new course.

During this experiment the QSM tools were examined. From the beginning SLIM Estimate proved added value to software development projects. We were able to generate scenario's with only few input. These scenario's were used to negotiate with the customer and to organise the project. Now we are able to make a founded offer to the customer based on experiences from the past. A small disadvantage was the fact that the experiences from the past were not coming from our own organisation but from projects from other organisations. Therefor it is important to gather own metrics as soon as possible. An other advantage of organising this tool was the improving of the planning process. The maturity of this process improved automatically.

The added value of SLIM Control was low during the first pilot project. There were two reasons for this. First of all the application was rather small (200 function points, 1100 man hours). The project manager was able to track this project 'by hand' and didn't need to use an advanced tool like SLIM Control. The added value of this tool will probably be higher with larger and more complex projects. The second reason was the fact that we were not able to track progress on realised lines of code although this is one of then main advantages of the tool. During the first pilot project we couldn't get the right data about realised functionality. This was caused by failing procedures and absence of tools to count lines of code.

During the second pilot project the added value of SLIM Control was much higher. The procedures to track realised functionality were adapted and worked. This project was also larger then the first one (290 function points, 2500 man hour). There are still some area's to improve like using these tools in a maintenance environment.

Lessons learned

You must have management commitment to have a successful improvement experiment. If you don't have this, the experiment is doomed to fail.

Using the tools stimulated the improvement of the process. The tools demanded a structured way of planning, tracking and using metrics. So the tools catalyses the improvement of the process.

Another important advantage is the generated information. This information is very useful in negotiating with the customer. You visualise the project and the consequences of certain decisions. This way you take emotion out of the negotiation.

Projects with an average size like in this experiment are too small to track with an advanced tool like SLIM Control. We think it is useful to projects larger then 2000 man-hours. The smaller projects can be tracked 'by hand'.

Disseminating and informing employees about the results and stimulate them thinking about improvements of the project can help implementing these improvements. Employees must see the added value of an activity. Only then it will be accepted.

The tools are based on 4000 business projects. This is useful to start with. It is better to have your own metric database to work with. This way the information you get from the tools will be more accurate.

TIP!
Start organising the software process for a solid environment to use the tools in
Get management commitment
Start collecting metrics right now!
Project monitoring office support projects actively to get commitment
Negotiate and communicate with the customer about concrete SLIM information

Result

As a result of this experiment Centraal Beheer now implemented and uses a metric program. The project monitoring office is operational and has added value in supporting the software development projects. The tool SLIM Estimate has to be used in making an offer to the customer. SLIM Control is an effective tool to track progress and to forecast a project. Projects must be at least above 2000 man hour to prove it’s added value.

Thanks to the subsidy of the European Commission we were able to improve the area’s of projectplanning and projecttracking to level 2 of CMM!

CV author Willem Bos

Willem Bos (willem.a.bos@achmea.nl) is process manager at Centraal Beheer. He is 28 years old. He graduated from a business school of informatics at 1996. During the first period he worked as quality manager in several projects. One of the most important projects was getting all information systems of Centraal Beheer 2000-compliënt. After this project he was involved in this experiment Plan-IT as process manager, responsible for the quality of the software process. His main focus was getting the project monitoring office started. He encouraged the use of metrics in the planning and tracking process. The metric program is implemented and growing. He is operationally involved in projects, doing function point analysis, supporting in planning and tracking a project. Within Plan-IT he is also responsible for disseminating the experiment results.

CV author Henry Meutstege

Henry Meutstege (henry.meutstege@achmea.nl) is project manager at Centraal Beheer. Henry is 34 years old, married and has two sons. He graduated from a business school of economics at 1988, where he followed a specialisation in business informatics. He worked in the Information Technology as programmer, systems analyst, information analyst and the last few years as a project leader. He worked for two software houses in the Netherlands, and started working at Centraal Beheer in 1996. Starting in June of 1998 he became responsible for the software improvement in the software development department he was working in. Having a large experience in software development he is considered the ideal person to create the bridge between the large amount of available theory, and getting it really done in practice.

He is the project manager in charge of Plan-IT. He made the management aware of the importance of a good metrics program and did a lot of good work in creating more understanding between the customers of the software development department and the developers.

Address

Centraal Beheer
PO Box 9130
7300HS Apeldoorn

References

M. Imai; *Kaizen : The key to Japan's competitive success*; McGraw-Hill, 1986.

M. Paulk, C.V. Weber, B. Curtis, M.B. Chrissis; *Capability Maturity Model for Software*, version 1.1, Software Engineering Institute; CMU/SEI-93-TR-24, February 1993.

Lawrence H. Putnam, Ware Myers; *Executive briefing : Controlling software development*; IEEE Computer Society Press

Lawrence H. Putnam, Ware Myers; *Industrial Strength Software, effective management using measurement*; IEEE Computer Society Press

Eliyahy M. Goldrath, Jeff Cox; *Het doel, een process van voortdurende verbetering*; Het Spectrum

Robert B. Grady; *Practical software metrics for project management and process improvement*; Hewlett-Packard professional books.

METRANET:

An extranet for the improvement of the Outsourcing of Software Maintenance projects using Function Points Analysis

Miguel Angel Martínez
Atos ODS S.A. Departamento Ingeniería y Servicios

Isabel Fernández
Atos ODS S.A. Departamento Ingeniería y Servicios

Introduction

This article relates the results of a Processes Improvement Experiment carried out by the Spanish software house Atos ODS within the 4th Framework Programme of the European Commission. The main objective of the experiment was the drawing up of a metric model based on Function Points Analysis for the management of the Outsourcing of Software Maintenance projects. ATOS´s Outsourcing Unit was made up at the start of this experiment of around 70 people. At this moment, more than 130 people are working in it. The Bank of Spain, as well as two of the main Spanish private banking groups, Argentaria and Banco de Santander, are among Atos´clients in Outsourcing projects. Besides banking, we have Outsourcing clients in the public administration, and in the health and telecommunication sectors.

In that experiment, Internet technology has been applied in such a way that the metrics system is in a web-database accessible by different players with different degrees of visibility: the Atos´s quality department, the Atos's technical team, and the client. To do that , the metrics tool SPI-Web, marketed by the Spanish firm MIT, was used during this experiment. The underlying idea is that of the *extended company*, as is found in the philosophy of Extranets.

Starting Scenario

The origin of this experiment must be found in the results of a quality assessment carried out during 1996. This assessment of the software development process, which was funded also by the ESSI program of the European Commission, was performed using the BOOTSTRAP methodology.

The service provided by Atos ODS rests on a Service Level Agreement basis. To verify these agreements we had to collect basic metrics such as response time, number of errors, lines of code -LOC, etc., but BOOTSTRAP assessment revealed deficiencies in process measurement and in the project management and quality assurance processes. The processes to collect metrics were poorly standardised, deviation between estimated and real data was high, and metrics results were exclusively used to review the project status with the client, not as a way of improvement.

During the contract definition phase Atos takes the compromise of maintaining a portfolio of applications guaranteeing a fixed level of quality. The maximum time accepted to solve possible errors is also fixed. The client assures a minimum level of service engagement.

Therefore it is essential that ATOS may collect data regarding the Service Levels fixed in the contract to review periodically the project results with the client. Data are also needed to decide when the contract agreements should be modified due to volume changes in the project .

Projects are mainly carried out at the client's offices by Atos' project team. The team is formed by the project leader, the analysts and the programmers. The client designs a Project Manager who has the responsibility of defining new tasks; the effort approval; and monitoring the project. The users are also involved requiring improvements of the systems.

This organisation is reinforced by the managers of both partners who monthly perform the monitoring of the service.

To assure the quality of the service, Atos has an specific Quality Department that looks after the use of standards. This team also measure the Agreement Service Levels established in the contract.

As a result of the BOOSTRAP Assessment, Atos has started this experiment with the following objectives:

The selection of a technical infrastructure to solve the problem that metrics must be recorded in the client’s facilities but should be visible from ATOS. The objective was to create a valid infrastructure so that the data were visible from ATOS and also available to the clients, so that every ATOS Director (and even the clients) can access the metrics from everywhere. This will allow them to know the project’s data as well as the set of projects and customers, and this will allow project benchmarking so that not only the Technical Department can initiate improvement actions (with regard to methods and tools), but also the Marketing Department can take its own marketing decisions.

The definition and the introduction of a Metrics Model for Management based on Function Points, aiming to increase the visibility of the Directors of the Technology, Marketing and Quality departments with regards to the Productivity and Quality levels of the projects.

From the start of the experiment we were conscious that to attain these objectives was difficult for different reasons:

The big size of the Information Systems: more than the 80% of the technical staff of our clients is employed in Maintenance activities.

The criticality of the service: the time solving errors is essential because most of applications are in use.

The Roles diversity: we know that the use of the new metric model and its acceptation at all staff levels could be a task for the medium term (1 year) because of the existence of different profiles with different needs and point of views. Then a lot of intensive training and awareness is needed to be able to introduce such models.

Plans and expected outcome

The initial step was the definition of the metric model, with the intention of checking, and evaluating cost, effort and quality of service. SIP, a Spanish consultancy firm highly experienced in quality issues, has been acting as a subcontractor, helping us to define and operate the standard set of metrics defined.

The first task was the *Study of ATOS needs.* Closely related to this study, an *status study* was performed. The current status of ATOS was analysed taking the baseline project as a reference, identifying the currently available information and the processes followed for gathering data. This work had a positive result confirming that it will be possible to gather all the model basic metrics in a short time. During the model definition phases, participation of directors was essential because they are the end users of those metrics, together with the clients, the project or service leaders, and the maintenance team..

Finally the metric tree was organised in two sets of metrics shown below according to the maintenance activities classification given by IEEE 1219 1993 : Corrective, Adaptive, Perfective and Preventive. See IEEE 1993 [1]. The first set of metrics is the Corrective / Preventive metrics set. The second refers to Perfective / Adaptive maintenance. This two set of metrics fits with the two main kind of activities performed and measured in our service level agreements:
Elimination of blocked situations (errors) – first set of metrics
Improvement of the system (new requirements) – second set of metrics.

Corrective-Preventive Metrics				
Name	**Metric Formula**	**Frequency of Reporting**	**Audience Level ***	**Scope**
PRODUCTIVITY				
Maintenance Coefficient	IMA = Maintenance Effort (in hours) / Application Size (in FP's)	Variable	CMM, ARM, CML, APL	Portfolio
Maintenance Productivity	PMA = Application Size (in FP) / Maintenance Effort (in hours)	Variable	CMM, ARM, CML, APL	Portfolio / Application
Application Support Trend	TAS = Support effort (in hours)/ Portfolio size (in PFs)	Annually	CMM, CML, CPL, ARM, AMM	Portfolio
Assignment Scope	AA = Application Size (in FP) / Number of maintenance staff (in persons)	Quarterly	ACM, AMM, APL, T	Portfolio / Application

QUALITY				
Reliability	IE = Errors number / Size of the application in Function Points	Monthly	All levels of the organisation	Application
Mean Repairing Time Repairing Time Deviation	TMRC = Time spent in correcting errors during maintenance / number of corrected errors DRPC = $= \sum(TRC_i - TRC) / N$	Monthly	CMM, CML, CPL, ARM, AMM, APL	Application
FINANCIAL				
FP Maintenance Cost	CM = [(Maintenance Effort in hours * cost) + other costs)] / Application Size in PFs	Annually	CMM, CML, CPL, APL	Portfolio
Corrective Total Cost	CR = (Maintenance Effort in hours * cost) / FP's	Half-yearly	CML, CPL, ACM, CML, APL	Portfolio / Application

Fig: Martin 1: Corrective Preventive Metrics

* Acronymes of the different roles played by clients and Atos' technical team and staff in Outsourcing projects: CMM Client´s Maintenance Manager; GML Client´s Maintenance Leader ; CPL Client´s Project Leader , ACM, Atos´s Commercial Manager AQM Atos´s Quality Manager, AMM Atos´s Metrics Responsible , APL Atos´s Project Leader, T Technician, ...

Adaptive-Precfective Metrics				
Name	**Metric Formula**	**Frequency of Reporting**	**Audience Level**	**Scope**
PRODUCTIVITY				
Enhancement Productivity	PM = Total size of the application portfolio after the enhancement / Total enhancement effort (in hours)	After delivery and acceptance by the client of the improvement project	CMM, CML, CPL, ACM	For a portfolio or for an individual application
Enhancement Delivery Rate	RM = Total size of the application portfolio after the enhancement / Enhancement Elapsed time (in hours)	At the end of each improvement project	CML, ARM, AMM, APL	For an application
QUALITY				
Stability Coefficient	IEA = (number of changes/ FP of the Enhancement)	Weekly for the 90 days after the installation of the application	CMM, CML, CPL, ACM, ARM, AMM	For an individual application
Error detection index *	IDE = (Nr. of errors in phase / Nr. of total errors) / FP's project or application	At least annually	CMM, CML, CPL, ACM, APL, T	For an individual application
Reliability	FA = 1 - (Application failures/ FP's)	Monthly or quarterly	CML, ARM, AMM	For an individual application
Testing Efficiency *	IHT = Nr. Of errors detected in tests/ Total Nr. Of errors	One month after operation	CMM, ARM, AMM, APL	For a portfolio or for an individual application
FINANCIAL				
Improvement Total Cost	CP = [(Effort used in the project * cost) + others)] / FP's	At the end of the project	CMM, CML, CPL, ARM	For a portfolio

Fig: Martin2. Adaptive, Perfective Metrics

* These metrics where not recorded during the project but keeping in the model .

In a second step, all the staff in the management level received several courses. Courses introducing FP helped them to understand a new way of talking about the size of a problem. The stress was put on the idea of talking the same language as our clients: Functionality. The courses also had several hands-on sessions where the participants learned how to count function points. Another set of courses, more focussed on the use of metrics for SW management, took place during this period.

The third step was the counting of the FP. The baseline project was an Outsourcing project running from 1995 for one of the main Spanish banking groups in a technical platform of COBOL-CICS DB2. This project deals with financial transactions that amount to around 180 million Euros daily and has a large number of interfaces with the rest of the bank's applications, so a serious error can have fatal consequences for the Bank's IT.

At it is known, FP counting relies on functional specifications but often legacy systems are not very well documented and specifications are out of date. During the experiment we had to face problems arising from lacks on documentation availability and its update level. We instrumented a set of directives to overcome them:

Systematic documentation and changes history checking.
The establishment of an efficient protocol for acquiring knowledge from the analysts of the applications to identify the key questions for the count.
Code diving, specially for the batch part of the applications.
The use of proprietary tools for static code analysis. The output of these tools provides information like database tables accessed, output reports, maps (screen forms) and allows us to draw a rough cross references schema. Cost derived from code diving was reduced in this way.

The baseline project includes 10 applications being maintained by the same team. The total size in lines of code -LOC- is near 2,000,000. LOC are counted using the well known definition of Hewlett-Packard: every program sentence, except comments and white lines. Among the project portfolios, the Banking Transfer application was selected to perform the initial FP counting. This application has a size of 464.000 lines of code and it is formed by 460 programs. The following table summarizes the main counting results: Final results where 749.07 adjusted FP for 464 KLOC of Cobol and DB2 code.

ILF's	EIF's	EI's	EO's	EQ's	Unadjusted Total FP
290	180	136	113	142	861

To benchmark the Banking transfer application with the rest of the portfolio applications we used the backfiring technique. Backfiring suggests rough measures of FP based on Lines of Code amount (105/107 LOC per FP in COBOL Language), see Jones [2].

Finally the Gathering of metrics using the SPI- Web Tool has started. As explained, the metrics Data Base is updated through the Internet from the client's premises. The data are visible on-line both by the client and by the management of Atos's Outsourcing Department.

With the SPI Web we are able to:

Capture the basic metrics of effort, cost, errors and number of changes used to derive quality and productivity metrics.

Automate the mathematical computing of Function Points from the basic Function Types.
Produce different presentations of the resulting metrics using a variety of histograms.
Define different user profiles with selective permissions to visualize the project data.

A web demo of this tool is accessible at http://Metranet.atos-ods.com

The implementation of the Improvement actions

By means of these metrics, trends in the reliability and maintainability of the applications and their cost and productivity ranges were analyzed and improvement measures took place. For trend analysis, we followed an iterative approach. The first results review shown below covers the january-april period. Below we shows the most significative results and improvement actions:

Maintenance Coeficient. (hour/PF)

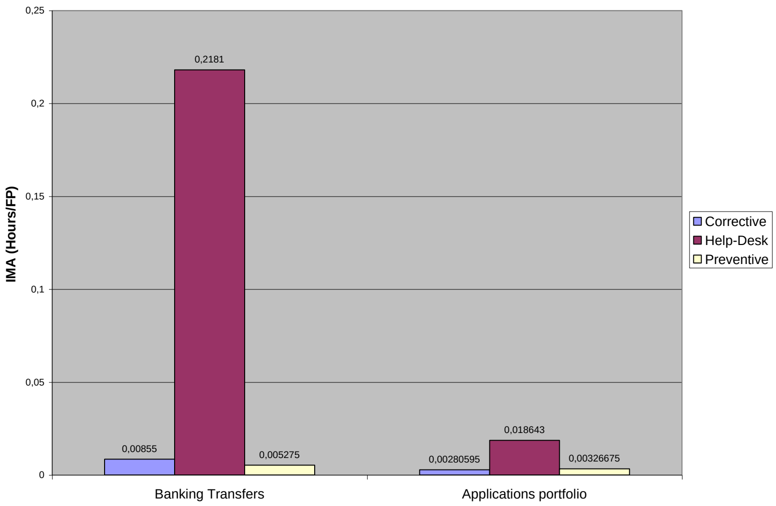


Fig. Martin3: Maintenance Coefficient Average. January-April period

Studying the Maintenance Coefficient (IMA) we analysed the effort spent in each maintenance category. The IMA for Help-Desk was bigger than for any other category. For Banking Transfer this metric was 10 times bigger than the average of the application portfolio. We think that this is due to a lack of user training in the use of the transfer application. The data confirm the general appreciation mentioned at the beginning: the application has a high degree of complexity.

The proposed improvement actions were:

Suggesting to the client specific user training courses
Orienting the responses given by the technical personnel not only to the immediate elimination of blocking situations raised in the consultancy by the user, but to understanding and familiarisation of the user with the use of the application.

This was the most critical situation detected because its impact on productivity is high.

Unfortunately on the second period there where not substantial changes in the trend of this metric. Suggested courses doesn't fit in the client's training plans for this year. They will be included in the next year training schedule.

Alternative proposition about giving elaborated and formative responses to users due to Atos ODS has not produced appreciable improvements. We think it is too soon to appreciate better results in this trend.

Maintenance Productivity (PF/hours).

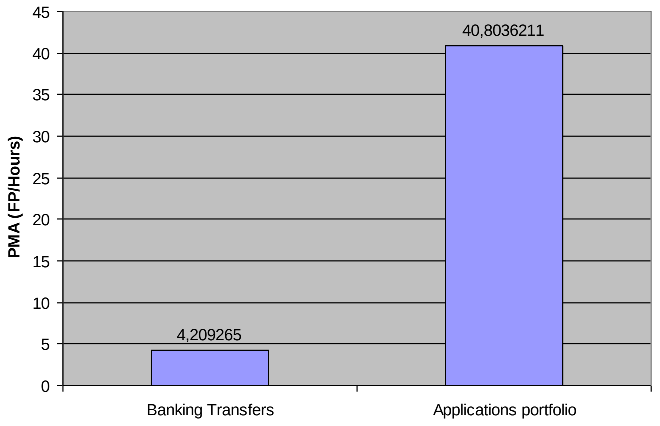


Fig: Martin4. Maintenance Productivity Jan April period

In the first period Mean Maintenance Productivity for the Transfers application was far below the overall productivity of the applications portfolio. As we saw, factors implied in this metric are closely related to Maintenance Coefficient metric. This fact explains that there are not significant changes in IMA behaviour either on the second period.

Reliability.(Nr.Errors/PF)

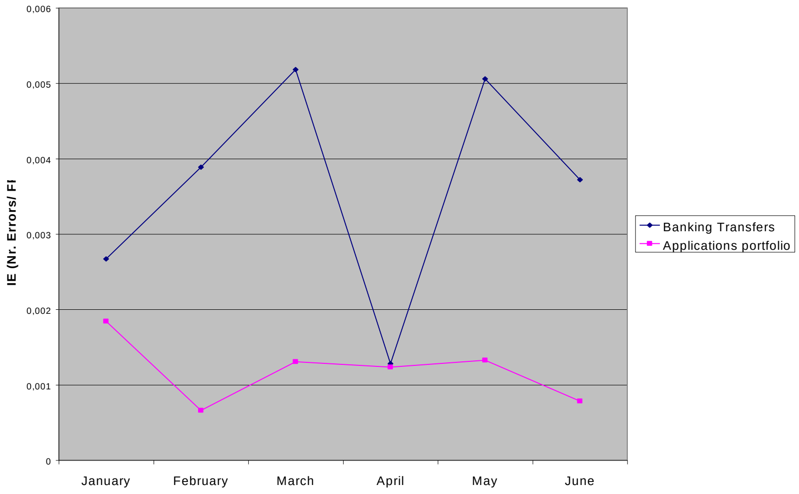


Fig Martin5: Reliability. Jan-June period

Studying the Reliability results we stated that the number of errors per FP that arise in the application of Transfers was greater than that in the rest of the applications. The immediate conclusion is the need to reduce this number of errors. In order to be able to take measures in this direction, it was considered necessary to collect additional information so that the errors could be classified by type and origin because initially errors where exclusively classified in terms of severity: severe errors are those that have direct impact in client's business & image. This was the objective set for the second review of results.

The collected Typology of Errors are shown below for the whole applications portfolio and for isolated Banking Transfers application:

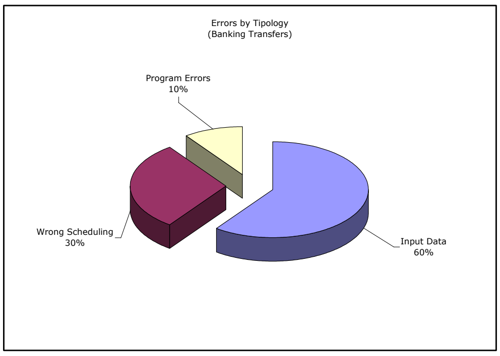


Fig: Martin 6. Typology of Errors Analysis for Banking Transfer

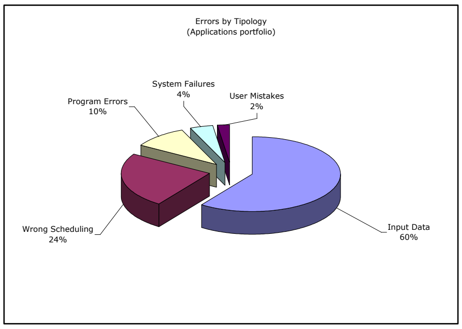


Fig: Martin 7. Typology of Errors Analysis for the whole project portfolio

This information has been essential to understand the evolution of Reliability metric.

60% of errors are due to bad Input Data. Data came from internal and external (between banks) operations. A first sight solution would be filtering data, rejecting erroneous registers. Nevertheless, this solution is not applicable by the following reason : external operations not processed means money loss (interests, breach of terms). Solution to this typology fall out of the scope field of Atos ODS.

30% of errors are due to Wrong Scheduling of installation processes once the software enhancements has been finished and the customer has received the system. This situation has been reported to the client. Solution to this typology fall out of the scope field of Atos ODS.

10% of errors are due to Program Failures. Atos ODS was developing improvement actions at this scope. This failures could be a consequence of the new code added to the application. Nevertheless, the low influence of this typology in the overall number of errors makes this actions insufficient to appreciate visible improvements.

The lower April values for IE are due to the less number of maintenance enhancements developed. This fact are related to Holy Week vacational period. That’s why the number of Wrong Schedulling errors has decreased. The same applies to Input Data errors: Minor number of enhancements has been developed by other applications, so few number of changes are done and few new interfaces with Banking Transfers has been added or modified. Collected data verifies those facts: April had only one error and its typology was “Input Data”.

Mean Reparation Time.

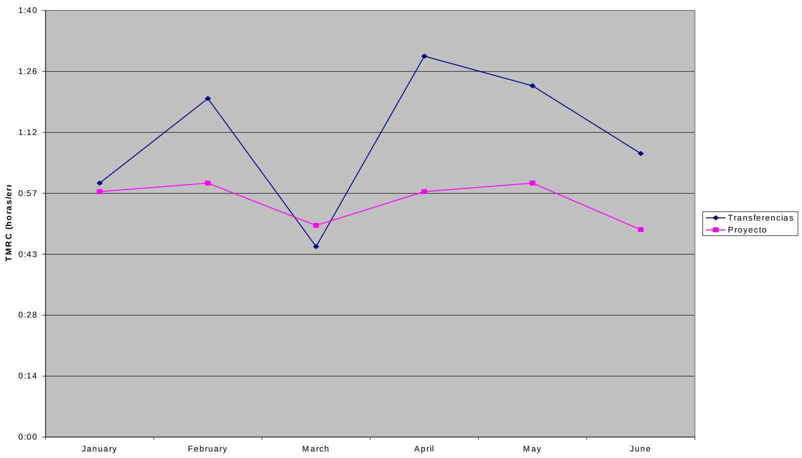


Fig. Martin8. Mean Reparation Time . Jan - June period

The Mean Repair Time to Critical anomalies (TMRC) for all applications under outsourcing is a metric used by the client to evaluate the service quality offered by Atos ODS. This graph shows that the MRTC of the Transfer Application is, in general, considerably greater than that of the other applications of the portfolio. The conclusion, then, is that the TMRCs of the overall application portfolio are clearly influenced by those of the Transfer Application, and that control and reduction of these can lead to considerable overall improvement

The reduction in the number of errors could be a way to improve repair times (greater number of available human resources and possibility of optimisation of tasks). Moreover, the possibility of a database with different search criteria of a historic portfolio of errors with information on the actions performed to solve them could reduce efforts in the work of analysis, above all in the case of similar or repetitive errors.

Although Banking Transfers TMRC is higher than the portfolio TMRC as a whole, Mean Repair Time is less than the maximum expected response time set by the client for solving severe errors.

The proposed improvement action where:

To create an historic data base of errors with information on actions to solve them.
To perform a SLA agreement review, reducing the maximum expected response time for severe and non-severe errors. This way, we expect to increased the client satisfaction and we to acquired a better status as services provider in the client organisation.

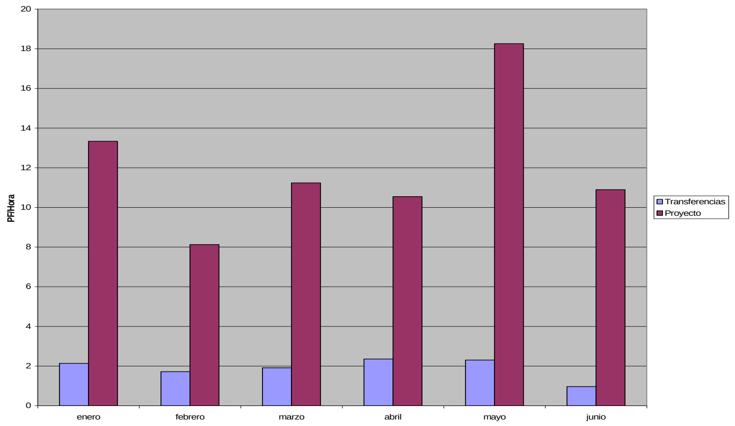


Fig. Martin9.Improvement Productivity. Jan - June period

Improvement Productivity

The results of the Improvement Productivity metrics were also worst than the rest of the portfolio applications. Nevertheless Banking Transfer analysts are experimented and have a good knowledge of the application, then we have considered it was due to the complexity of the new improvements and we do not propose any improvement action, looking forward the next review.

Nevertheless the analysis of the second period results help us to understand the real source of the problem. In June there was more than a 100% of effort increase for only a functionality increase of seven Function Points. In this month many effort was spent in adaptive improvements.

Fiability of the Improvement

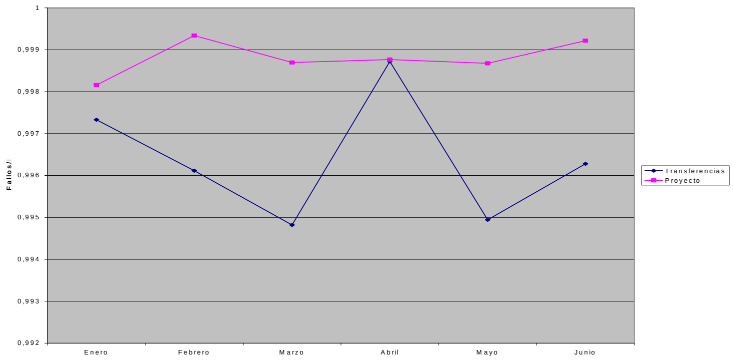


Fig. Martin10. Improvement Fiability . Jan - June period

To improve this result the proposed improvement action were the same than those explained in the Corrective Fiability.

The measured results and the lessons learned

The incorporation of Function Points (FP) as one of the basic metrics of the metric model was carried out with the following aims:

Standardisation: To check the quality of the applications and the productivity of the teams using a common language based on an international standard.
Quality Improvement: To improve the quality of the service, benchmarking projects and allowing management to undertake improvement actions.
Management Cost Reduction: To reduce the time and effort employed in management of outsourcing.

Taking into account previous results we estimate that at the end of the PIE, the majority of those objectives had being attained .

Standarization: The FP counting is quite unambiguous in such a way that -despite the interpretability of the technique in some aspects- finally the same number of FP are obtained when the counting are performed simultanesly by diferent persons (external consultants or Atos technicians).

Quality Improvement: The trend analysis is too short to evaluate significant improvements (we estimate that more than a year is needed to obtain significant results). But benchmarking of applications has been proved as an useful technique and many improvement actions have been launched as a consequence.

Management Cost Reduction: The availability of the Internet and the use of the SPI-web meant a significant decrease of management effort by highly simplifying the elaboration of reports and the preparation of the review meetings with the client where the client receives a report with the most relevant figures of SLA (number of errors solved, number of enhancements implemented, cost/effort spent ...). The production of the quaterly reports takes betwen 20-25 man days. With SPI-web we could prepare a complete dossier in less than 7 man days.

Metric	First Review			Final Review			Positive trend	Trend In the period
Corrective								
Maintenance Coefficient	0.008	0.21	0.005	0.09	0.22	0.005	Lower values	=
Maintenance Productivity	4.2			4.2			Higher values	=
Application Support Trend	0.226			0.233			Lower values	-
Assignment Scope	192.12			194.62			Higher values	+
Reliability	0.002			0.003			Lower values	-
Mean Repair Time	1.30			1.07			Lower values	+
Repairing Time Deviations (Critical Errors)	-3.95			-3.86			Lower values	+
FP Maintenance Cost	2018.5			2178.4			Lower Values	-
Corrective Total Cost (Repairing cost)	6034.4			9146,4			Lower Values	-
Improvement								
Enhancement Productivity	2.3			0.9			Higher values	-
Enhancement Delivery Rate	8			9,5			Higher values	+
Stability Coefficient	0.1515			0.2413			Lower Values	-
Reliability	0.998			0.995			Higher values	-
Improvement Total Cost	3316,6			7370,4			Lower values	-

Fig: Martin 11. Final Results

Lessons Learned

Impact of the state of the documentation in the quality of FP counting. When Functional specifications are low quality, or not well updated, then the FP counting process is very hard and time consuming. Alternative solutions must be applied in this situations (we think it must evolve in the style of automated assisting counting process). A future measure to improve the service is to suggest and negotiate with our clients in the very earlier phases of the projects the minimal documentation set to be maintained to allow the FP counting.

Difficulties interpreting the IFPUG rules. Function Points is a well accepted size measure for SW applications, but the IFPUG counting rules are not always easy to apply. Different people could obtain different results measuring the same applications. Our goal is to apply FP always in a consistent manner inside the organisation and the best way to achieve this is to create a group specialised in the counting process with clear norms and procedures that be always the responsible for the counting.

We realised that adopting Function Points Analysis as the standard for the Outsourcing of SW Maintenance Service means, from Atos' point of view, important changes in procedures and management of projects. From the customer's point of view, important changes will have to be carried out in their own organisations' processes for undertaking an outsourcing project. Besides, they have to add the cost of familiarising themselves with the essential concepts involved in Function Points Analysis. Providers should explain to the client that the additional effort needed to work with metrics is largely justified because they are an excellent way to obatin a better service.

The greatest benefits from the use of Function Points for the Outsourcing of Maintenance should be expected not so much in the estimation of maintenance activities (only a small subset involves change in functionality) but in its capacity for clear contractual identification of the service commitments acquired and in its potential to allow benchmarking of projects as an instrument to undertake improvement actions.

When FP are applied to Software maintenance there is a risk of obtaining flat results. Perfective actions are suddenly very small, then it could be very difficult to distinguish one from another in terms of FP (we have obtained the same results in the 80% of the 20 perfective requests evaluated in Metranet: 5 FP). FP variants as Micro Función Points with a high granularity level could be very useful to avoid this issue.

Improvement through metrics is a long term project. In this sense, Atos has the purpose to keep in use the metric model with the following aims:

To improve the way we work with the client through the use of metrics. Even if that implies an awareness effort, we think it is the only way of obtaining the total confidence of the client that we need to evolve through Total Outsourcing.

To perform a depth analysis of the productivity and quality deviations obtained for every application considering new improvement actions based in the use of new tools and the start of specific training courses.

To improve the management of resources and projects using internet technology. The on-line analysis of the project data will allow to react immediately putting in place the needed actions.

Appendix 1: Authors

Miguel Angel Martínez Jimeno

Graduated in Physical Chemistry from the Universidad Autónoma de Madrid (1989) where he worked as an Associate Professor in the Department of Physical Chemistry (90-91).
Teacher of the doctorate course "Applied computing for topographers" (April 1994) - Colegio Profesional de Topógrafos de Madrid. In the field of Software Maintenance, he has directed the development of the CONDUCT tool for the management of maintenance projects and is co-author of the book: Piattini et al. "Mantenimiento del Software Conceptos, métodos, herramientas y outsourcing" (Software Maintenance, Concepts, Methods, Tools and Outsourcing). Pub. Ra-ma. Madrid. 1998. At the present time he is responsible for the implantation of quality plans in the projects of software maintenance.

Isabel Fernández Peñuelas.

EDUCATION 1983: B.A. in Philosophy from the Universidad Complutense ("UCM"), Madrid, Spain. 1984: Masters in Philosophy of Natural Language. 1985-1986: Doctorate Courses at the Department of Logic and Philosophy of Sciences. 1985-1987: Scholarship from the Spanish Ministry of Education at the UCM. 1986: Assistant Professor in the Department of Philosophy, UCM. Practical training in PROLOG II Language.

R&D PROJECTS 1985-1987: "Computatio Orteguiana" Computerized Lexical study of the Ortega y Gasset Corpus. 1988-1992: Knowledge Engineer of the Metapedia Project; Expert System using Object Oriented technologies for the design, storage, and edition of the ESPASA-CALPE Encyclopaedia. 1992-1993: PONTIFEX Project; Objective: Object-Oriented decision support framework for handling routing and scheduling problems (ESPRIT II n. 2111). 1995: Chief of the project SPAS; Assessment of the Software Process Development of the Mainframe Unit of Sligos S.I. using BOOSTRAP methodology (ESPRIT IV n. 21.374). 1996: MANTEMA Definition of a Software Maintenance Methodology (R&D) project sponsored by the Spanish Ministry of Industry & Energy).

Appendix 2: Organisation Description

Atos Group is one of the main participants in the world of European services companies with more than 9,500 employees and a turnover for the 1997-1998 financial year of 957.4 million Euros. The implantation by sectors is shown in the following division of activities: 43% in banking, finances and insurance; 21% in industry, automotive, aerospace and transport; 16% in telecommunications and media; 13% in distribution; 7% in other sectors.

Atos ODS S.A. is the company within the Group that carries out its activity on the Iberian Peninsula. The territorial implantation on the Iberian Peninsula includes the following centres: Barcelona, Madrid, Lisbon, Valencia, Valladolid, Zaragoza and Andorra. During the 1997-1998 financial year, it achieved an invoicing of 6,864 million Pesetas, for activities with 550 clients and a human team of 950 persons.

Since its creation, Atos ODS has been characterised by being a company that bases its service on the application of technological, functional and business knowledge in areas that require a high degree of specialisation. The use of a strongly technological component in its projects, together with the functional component, make the natural market for Atos ODS the one where specialisation is indispensable.

Atos ODS has a wide offer of products and activities : Engineering Services (Outsourcing, Systems reengineering, Consulting, and systems development using advanced technology).
Products for Real Time environments (real time operative systems; ADA compilers, development kits to embedded software appliances); Telecommunications (telecommunications management network, intelligent networks, GSM ..); Industry (products and services in the CAD/CAE market: aeronautics / aerospace, automotive, machinery, transport; consultancy and implementation of SAP systems for enterprise management, business process reengineering service , ERP Solutions); and Technology (workflow , network and systems management tools, middleware and distribution, software for connectivity and communications).

References

[1] IEEE, std 1219: Standard for Software Maintenance. IEEE Computer Society Press. USA, 1993.

[2] Jones, C. Applied Software Measurement. McGraw-Hill, 1991 (ver si es 1991 1996)

Bibliography

Sorrentino, M. De Gregori, G., La Manutenzione del Software Applicativo. Ed. Franco Ageli , Italy, 1995.

Pigoski, Thomas M., Practical Software Maintenance. Best Practices for Managing Your Software Investment. John Wiley & Sons, 1997.

Piattini M., Villaba J., Ruiz F, Fernández I, Polo M, Bastanchury T., Martínez M.A.,Mantenimiento del Software. Conceptos, métodos, herramientas y outsourcing, Editorial RA-MA, Madrid, Julio 1998

Kuvaja P., Smila J. Krzanilk L., Bicego A., Saukkonen S., Koch G., Software Process Assessment & Improvement. The Boostrap Approach. Blackwell Publisher, 1994

Abran, A. Nguyenkim, H., "Analysis of Maintenance Work Categories through Measurement". Proceedings of the International Conference on Software Maintenance. IEEE Computer Society, 1991

Fenton N.E., Pfleeger S.L., Software Metrics. A rigorous & practical approach, International Thomson Computer Press, 1997

Albrecht , A.J., Measuring application development productivity. Proceedings of the IBM application development symposium, Proceedings of the Joint SHARE/GUIDE ,Monterrey, Canada, oct. 1979 pp. 83-92

IFPUG, Function Point Counting Practices Manual Release 4.0, IFPUG Standards , 1994

IFPUG, Guidelines to Software Measurement Release 1.0, IFPUG Standards, 1994

IFPUG, Function Points as Asset - Reporting to Management, IFPUG Standards , 1992

Abran A., Maya M., A Sizing Mesure for Adaptative Maintenance Work Products International Conference on Software Maintenance, ICSM-95, Opio, France, 1995

St-Pierre D., Maya M., Abran A., Desharnais J-M., Bourque P., Full Function Points: Function Points Extension for Real-Time Software- Counting Practices Manual, Université du Québec à Montréal, Montréal, Technical Report no. 1997-04, September.

Jones C., A Short History of Function Points and Feature Points, Software Productivity Research Incorporated, mimeo version, 2.0, Feb. 20, 1988

Software Metrics for Process Improvement Experiments

Terttu Orci

Royal Institute of Technology
Stockholm University
SISU
Stockholm, Sweden

Introduction

Software process improvement (SPI) seems to be a commonly agreed silver bullet to solve the software crisis. The underlying assumption is that a good software process results in a high quality product, delivered in time and within budget. All SPI actions have one or several of the goals: to maximise the product quality, and to minimise the time and the cost. An organisation initiating SPI work can choose among publicly available models for the purpose, e.g. CMM [15], Bootstrap [12] and SPICE [18], or locally designed actions can be initiated in areas where improvement is needed. Success stories and implementations of SPI are published, e.g. in [1], [8], sometimes with impressing figures in financial terms of the return on investment. More and more organisations initiate some kind of SPI actions.

The European Commission supports a particular programme called ESSI - European Systems and Software Initiative, to encourage the European software organisations to undertake process improvement experiments (PIEs) [3]. An obvious value of the ESSI programme for the European software market is the sum of the values for the participating organisations. However, an additional value should be obtained through so called dissemination actions, spreading the experiences and lessons learnt, and preventing other organisations to invest in reinventing the wheel, or repeating the mistakes done by others.

The project EUREX, European Experience Exchange (No 24478) [2] is one of those dissemination actions. The objectives of EUREX are to collect, systematise, and disseminate the experiences and lessons learnt in the process improvement experiments. Naturally, there is a wide variety of PIEs along the dimensions of the subject domain, organisation type, working methodology, objectives of the experiment, to mention a few. The data collection in EUREX is by workshops, discussions in the context of the workshops and otherwise, and studies of the final reports.

The partner SISU (Swedish Institute for Systems Development) is responsible for the subject domain Software Metrics in the EUREX project work. In a way, all the PIEs should be classified into the subject domain software metrics, as every PIE should include measurement to determine the magnitude of the improvement. However, although software metrics is an established field of software engineering in theory and research, its industrial applications are behind. The PIEs are not any exception, as was pointed out in the study at EuroSPI98 [13]. Although the study was very limited, there was a clear indication of a major potential for the metrics maturity improvement in European software industry.

Measurement is the vehicle for control. Without that control, software engineering will not reach the status of sound engineering discipline, but remains a craft. There is a need to point out the importance of measurement, to discuss how it should be conducted and how it should not.

The Objectives of the Study

The objectives of this study are to increase the awareness of sound software metrics in general, and in particular to point out how a number of PIEs have handled the metrics related issues.

The framework for the study is software metrics theory and ESSI guidelines [4], described in Section 4 more in detail.

Outline

In Section 2, software process improvement including ESSI programme. Section 3 presents the basic issues of software metrics, both from theory and practical point of view. Section 4 presents the framework for the study and the results are presented in Section 5. Section 6 contains concluding remarks.

Software Process Improvement

Software process improvement is commonly agreed being the silver bullet to solve the software crisis of the organisations. It is argued that technology and people in an organisation change, whether we want it or not, but the processes, i.e. the way we do things, should be stable. The stability should not imply stagnation, but controlled change. An organisation with stable processes should regard software process improvement as a continuous, never ending activity.

Software process improvement can be studied from different aspects, e.g. the model for improvement used. In this paper, the context is process improvement experiments within ESSI programme, i.e. organisation wide processes, limiting out e.g. individual efforts undertaken by software engineers like PSP [9], [10].

SPI in the Context of ESSI

Some of the large software companies in USA have published their long term process improvement programs including introduction of metrics, e.g. [8],[16]. We are not aware of any corresponding publications from large European companies concerning organisation wide improvement or metrics programs. The closest we come in Europé is the ESSI programme and its publications. The PIEs with approved final reports are intended to be available on the world wide web [4]. The dissemination actions publishing aggregations and summing up of the experiences and lessons learnt are also publicly available as EC project results.

The PIE projects are normally short term, 12-18 months in duration, and intended to improve the software process of an organisation in some respect. If a project is successful, and welcomed by the staff, it may very well generate further improvement actions. If it is a failure, or if the staff does not see the gain for their working situation or business as a whole, further improvement actions may be hard to propose and to get accepted.

Model for Improvement

Software process improvement may be undertaken based on a general, publicly available model, e.g. CMM [15], SPICE [18], Bootstrap [12], PSP [9], [10], ISO9001 [14], or it can be performed using a local model designed by the organisation for the purpose. The general models usually require a long-term investment and duration, longer than 18 months, which is the normal duration for a PIE. Therefore, the PIEs usually have a locally designed model, described in the work packages. The local model should ideally be extended by measurement, along the guidelines of ESSI.

The software process – whether new or changed - must be defined and documented. CMM or any other model can be used as a meta model, to show the way *what* should be in place. Every process on an adequate level of detail, should include the following:

goals
a commitment in terms of policy statements and leadership
abilities in terms of resources and training
a set of activities which may be plans, procedures, tasks, reviews
measurement of the status and variability from the plan
verification by management and software quality assurance (SQA) of implementation and institutionalising.

In addition to the processes, standards for the different work products may be defined, e.g. coding and document standards.

Within the PIE projects, there is naturally a management commitment for the activities, as ESSI programme only supports the half of the costs. The abilities in terms of resources is relevant within PIEs as well as in other software process improvement efforts. Experts can be used to cover a lack of competence in the organisation, for defining the process and in training the employees in the new process or in measurement related issues.

Measurement and analysis intends to determine the status of the new process, whether it is followed and is functioning as intended. Verification is the same type of control from the management perspective.

Software Metrics

Software metrics, presented in various textbooks, e.g. [6],[7],[10],[15] and conferences and workshops [7], has a long tradition in theory, while considerably shorter in terms of industrial applications. Software metrics relies on the underlying theory, called representational measurement theory, posing some requirements on a correct definition, validation, and use of software metrics. From practical point of view, there are several further questions of importance, e.g. how to identify the right metrics to use, how to introduce a metrics programme, and how to keep it alive.

Software measurement is an activity assigning a number or a symbol to an entity in order to characterise a property of the entity according to given rules. The informal definition, even though giving an idea, must be more precisely defined. The message of the definition is that there should be an entity, a property, a measurement mapping and rules for the mapping. The measurement mapping and the rules is usually called metric. An example of an entity is code. An attribute characterizing the code is size, and one possible metric for measuring size of code is the number of lines of code (LOC).

Initially, there must be an intuitive understanding of the property of the entity of interest, otherwise there is no way to define an adequate metric. For example, for the entity person, we can intuitively understand the property length, which can be measured in inches or centimetres. If observing two persons, we usually get an understanding who is taller, i.e. whose length would get a larger value if measured. The intuitive understanding can be represented in an empirical relation system, a pair consisting of the set of entities, and a set of relations, e.g. “taller than”. For the measurement, there must be a corresponding numerical relation system, a pair, with symbols representing the entities and numerical relations corresponding to the empirical relations. For the relation “taller than”, an adequate numerical relation would be >. There is also a so called representation condition requiring that a measurement mapping must map the entities into numbers and empirical relations into numerical relations in such a way that the empirical relations preserve and are preserved by the numerical relations. In practice, this means that if we have an intuitive understanding that A is taller than B, then also the measurement mapping M must give that M(A) > M(B). The other way around, if M(A) > M(B), then it must be that A is intuitively understood be taller than B.

The measurement mapping, the empirical and numerical relations are usually called the scale of the measurement. There are five different scales: nominal, ordinal, interval, ratio, and absolute scales. It is important to establish the scale of the measurement in that different scales allow different manipulations with the measurement data.

What is the thing being measured?

There are three main classes of entities of interest for measurement in software engineering, namely *product, process,* and *resource*. Product is an output from a process, e.g. code, a document, a script. A process is one or several activities. Resource is an input to a process, e.g. staff, tool, method. Sometimes, we need to measure attributes for a *global* entity, namely the entire organisation, e.g. average delivery delays in all the projects undertaken during a certain period of time, or the maturity of the organisation in software development on a CMM scale.

Unless there is a clear statement of the entity, attribute, and metric, it does not make much sense to talk about measurement. For example, the statement “the size is 20 measured in LOC” does not make sense unless we know the entity in question. Unless the attribute is defined, we do not know what property of the entity is supposed to be characterized by the metric. For example “The code has FOG number 50” does not make any sense unless we know what attribute we are measuring by the FOG number. Unless the metric is defined, we do not know even the scale of the measurement, nor can we get an understanding of the relative value of the measurement. For example, the statement “The code size is 70000” does not make sense unless we know if size has been measured in LOC or bytes, or something else.

The Study

The framework of the study is software metrics theory and ESSI guidelines for mid-term and final reports [4]. Below, those parts of the guidelines in some sense indicating measurement are given:

Objectives
Explain the method and specific metrics used to measure the impact on the business goals and to verify to what extent the problem has been solved.

Phases of experiment
Explain any specific training undertaken as part of the experiment as well as other internal dissemination activities.

Consultancy during the experiment
Explain the role of internal and external consultants, if any, on the project and the reasons why they were needed as well as the effort expended.

Resulting scenario
This section should detail the actual results obtained from the experiment and your analysis of them compared to the original objectives of the experiment. Include any qualitative and quantitative results and how they were measured. Provide details of any final assessment and, as far as possible, give real figures.

Key lessons learnt
This section should summarise the key lessons, positive and negative that you have leant to date from undertaking the experiment. It should identify clearly your key lessons leant, from the technological and business point of view.

The interpretation of the ESSI intentions we have made is that the expected impact on the business goals should be indicated in quantitative terms and the obtained impact should be measured and compared to the expected, and the metrics used should be reported. Further, specific training if applicable, should be indicated. The specific training may be training is software metrics. Consultants are sometimes needed in a PIE, they may be partners with the required skills and competence, or their services can be bought from an external organisation. In that case, it is interesting to investigate to what extent the universities have participated in the role of metrics expert. Key lessons learnt are important messages to other organisations planning process improvement experiments.

The following questions will be studied:

Have the objectives been described in quantitative terms?
Have any quantitative results been reported?
Has metrics specific training been conducted?
Has measurement related consultancy been needed?
What are the metrics related lessons learnt?

In addition to the above questions, we also investigate the distribution concerning the entity measured, i.e. process, product, resource, or global. Further, we present a set of measurements, which are well defined, and another set, which is unclear. The unclear measurements are discussed to some extent. Further, we also cover the question how large a percentage of the total number of measurements are well defined in the sense that all three of entity, attribute, and metric are defined or can be understood from the context.

In the following, we give the motivations for the different aspects under study.

Have the objectives been described in quantitative terms?

This aspect is directly related to the guidelines heading “Objectives”, which recommends to explain the method and specific metrics used to measure the impact. Although the objectives of the PIEs are widely varying, there is a common denominator, an improvement of a process.

If the objectives are expressed in quantitative terms, it indicates that measurement has been understood as a vehicle for determining the magnitude of the improvement already at the project proposal time.

Have any quantitative results been reported?

This aspect is directly related to the guidelines heading “Result”, stating that the results should be compared to the original objectives of the experiment. If the objectives have not been quantified, the results may or may not be quantified. It is possible that the good intention – formulated at the time of the application - to measure the results and check the fulfilment of the quantified goals may not been possible to fulfil. On the other hand, if the goals have not been quantified, but the results are expressed in quantitative terms, the project staff might have learnt the lesson during the project that quantification and measurement are important for various reasons, not least for the management to keep the commitment and to report to the EC.

Has metrics specific training been conducted?

The guidelines recommend to describe the specific training activities needed to conduct the improvement action, for example metrics training. If such a training has taken place, it witnesses of a policy to increase the metrics maturity in the organisation in the long run, not merely within the PIE project. Otherwise, external consultants may have been used for the sake of the project. A positive answer to the metrics training gives a better picture of the metrics awareness, indicating an investment in a long-term improvement.

Has measurement related consultancy been needed?

The guidelines include the heading “Role of consultants”. It is interesting to investigate to which degree the companies have needed expert guidance for measurement related issues. If experts have been used, it is interesting to investigate the type of the expert organisation, university, or consulting company.

Metrics related lessons learnt

The intention is that the lessons learnt should carry over from a PIE to other organisations planning software process improvement. However, the wide variety of PIEs along dimensions like organisation type and size, ways of work, maturity, subject domains, type of products or services provided, makes it difficult to utilise the experiences on any detailed level. However, there are some general lessons learnt, applicable to any organisation, that deserve repeating over and over again. Only lessons learnt, specific to measurement, will be considered here.

Are the measurements well defined?

A well-defined metrics is first of all characterised by the presence of all three of entity, attribute, and metric. If any of these is missing, there is no much point for measuring. Metrics must also be valid, which means that the representation condition is satisfied, i.e. the empirical relations preserve and are preserved by the numerical relations. That is, however, difficult to determine from the final reports, and therefore omitted from the study. We only consider if all of entity, attribute, and metric have been clearly stated or implicit in the context.

Results

We have studied ten PIEs along the aspects described in the previous section. The PIEs have been randomly selected from a larger population, and thereby may represent small and large organisations, different subject domains, and the like.

Have the objectives been described in quantitative terms?

Only one of the PIES (10%) has expressed the objectives in quantitative terms, giving exact figures: *10% reduction of error reports for software products, and 5% shorter time to market.*

All the other PIEs indicate the improvement in general terms. Below, a few examples of the objectives are presented, slightly syntactically edited without changing the semantics:

To improve the specifications
To increase the maturity level of a process
To improve the maintenance process
To establish a quality model for the full life cycle process
To improve software quality by testing
To improve design and development process
To experiment a methodology
To demonstrate the applicability of a method

The objectives are quite general, some indicating improvement, e.g. *to increase the maturity level of a process*, some only indicating to do something, *to introduce object-oriented technology*.

Have any quantitative results been reported?

80% of the PIEs have presented quantitative results. The rest of the PIEs have not measured anything at all. However, it is interesting to match the quantification of the results, i.e. how the results have been measured, to the objectives stated. It should be noted that only one of those eight PIEs presenting quantitative results, did express the objectives in quantitative terms.

In the following, we discuss a few examples of the measurements compared to the objectives stated.

To improve the specifications

There is no measurement at all in the PIE. It is easy to understand that it is difficult to measure the quality of the specifications. One way of measuring improvement of specifications would be to measure the number of specification errors discovered in the product, during development or after delivery. That requires logging of faults per development phase. Such data collection and analysis has not been reported.

To increase the maturity level of a process

The maturity level has been measured in terms of a maturity level according to a particular maturity model. In addition to that, several other measurements, not indicated in the objectives, have been used, e.g. *time* spent resolving problems related to the process, *efficiency* of the new process in terms of producing its deliveries. All the attributes describe the new process.

To improve the maintenance process

The improvement objectives are described in terms a number of subgoals, e.g. to track the problems and actions efficiently, to introduce defect prevention activities, to provide quality metrics, and to improve the effectiveness. The measurements undertaken are distribution of the efforts of maintenance in different activities, the distribution of the causes of problems per development phase, size and complexity metrics assessment. The real challenge is of course the introduction of defect prevention activities, which in fact has not been introduced, but prepared for by tracking the distribution of the causes of problems per development phase. The size and complexity metrics are LOC and McCabe.

To establish a quality model for the full life cycle process

The PIE indicating such ambitious objectives introduces a number of measurements in the organisation, concerning the attributes *productivity*, *cost*, and *quality* of a number of specific processes. Productivity is measured using function points in development process, and the productivity of processes is measured in terms of accomplishment of the process specific tasks. Quality of development is measured in terms of deviations from time and cost estimates, and the number of errors per function point. The quality of the other processes is measured in process specific terms.

To introduce a specific technology

The improvement of introducing a specific technology has been measured in number of workdays to produce a version of a product using the old and the new technology. The improvement is measured only in terms of *process efficiency*, which is an important measure. However, if also the quality of the product had been measured, e.g. the number of error reports, it would balance the measurement of the improvement to some extent.

Has metrics specific training been conducted?

20% of the PIEs report metrics training. There are of course several possible reasons for not training in metrics, e.g. the organisation considers the staff possessing enough metrics skills, or that there is no need for long-term investment in metrics, but only within the PIE.

Has measurement related consultancy been needed?

60% of the PIEs have used consultant for metrics purposes. In 50% of the cases, the consultant was a university. It is difficult to interpret anything into the figure 50% university consultants in metrics.

Metrics related lessons learnt

There are a large number of statements concerning measurement experiences. We present all the unique statements.

Reliable data was missing
Some results are subjective
Metrics to be collected should be simple
Only data that will be used, should be collected
Metrics should be collected at accurate frequencies
There should be clear currency/time limits
Administrative overhead should be reduced
Educate people to why data is needed
Automate collection of measurement data
Determine goals of the metrics program
Inform that metrics are collected
Provide feedback to those who collect
Measurement data is useful to know if you are going to right direction
Data collection is demanding
Results are useful for the commitments of the management and developers
Measurement required more time than planned
Existing data was not good
Data collection should be an integrated part of the work process
Data collection and analysis are time consuming

All the statements are common knowledge, there are no big surprises. It is obvious that data collection and analysis is time consuming, and should be automated as much as possible. Data collection must also be an integrated part of the work process, otherwise it is easy to forget or neglect it, as soon as the development activities are pressing. An important aspect is the quality of the data collected. Obviously, there has been some problems with quality as there is the statement of the importance of reliability, and observation that existing data was not good. A related issue is the currency/time limits, and accurate frequency. All the products measured should be under configuration control, otherwise the data is misleading and erroneous. It is important to motivate those who collect the data, by informing about the reasons, and by giving feedback from the analysis results. For some attributes, the best we can do is subjective judgements, e.g. the working satisfaction of the employees. The most important aspect of subjective estimates is to remember that they are subjective estimates, and not hard measurement data.

Are the measurements well defined?

There are totally 47 measurements reported. The distribution of the measurements is 45% process, 26% product, 11% resource, 6% global, and 13% with an unclear entity type.

In only 12 of 47 measurements, corresponding to 26%, all of entity, attribute, and metric were either defined, or clear from the context. The attribute was unclear or not stated at all in 55% of the measurements. In 21% of the measurements, metric was unclear or missing..

In the following, we give examples of measurements, which are well defined and examples of unclear measurements, and discuss the weaknesses and possible interpretations.

Examples of well defined measurements

Document management effort is a measurement of the attribute effort of the entity document management process. Effort is measured in person time of some granularity. *Productivity of development in FP/PM* measures the entity development process and the metric is calculated as the ratio of output in functions points and person months. *Cost of FP* measures the entity type product for some particular product, or it could be an average for the whole organisation. The attribute is cost, and the metric may be person hours, or it could be money. *The number of faults in code* measures the entity type product, for a specific code, the attribute is quality, and the metric is the number of occurrences, i.e. absolute scale measurement. The measurement is well defined in that it includes all the components of entity, attribute, and metric implicit or explicit. However, a more fair attribute would be defect density, calculated as ratio of the number of defects and the size of the product.

Examples of unclear measurements

Size is a usual attribute for the entity type product, e.g. code. The metric may be LOC if measuring code, or the number of pages of A4 format if the product is a document. However, even the seemingly simple metric LOC must be well defined, in terms of what lines are included and excluded, respectively. Unless such a precise definition is formulated and followed when measuring, the measurement is waste of time and comparisons misleading.

Test time delay requires some kind of interpretation. Possibly there are several. It may be a measurement with the entity type resource, which can be a person acting as an estimator of the test time. Test time delay would then be the difference in duration between the estimate and the actual, indicating the prediction capability of the estimator.

Test process efficiency is a measurement, obviously measuring the attribute efficiency of a specific test process. The metric is not given, however. Efficiency could be measured in the number of tests performed during a certain amount of time, or by the number of defects found during a certain period of time. In order to make any use of it, it should be an average of the test efficiency of a number of tests.

Professional awareness concerns an entity person of type resource, and should probably be an average of the professional people concerned. The metric is, however, missing. To our knowledge, there is no obvious metrics for awareness, but only subjective judgement. The same goes for *employee satisfaction*, *working motivation*, which have been stated by some of the PIEs.

Maintenance effort probably indicates some attribute of a product, for example simplicity, flexibility, maintenance friendliness, or an aggregated attribute involving them all. Without stating the attribute explicitly, it is unclear whether the metric is a valid metric at all, as the intuitive understanding of the attribute is not obvious.

Distribution of problems/product is a product measurement. It is probably intended to give an indication of which products require more effort than others, for the purpose to allocate more resources to future maintenance for an extremely demanding product. Problems must, however, be clearly defined, and also the handling of same problem occurring several times should be clearly separated from unique problem occurrences.

Conclusions

We have analysed ten PIEs with the focus on how the improvement has been measured. The basis for the analysis consists of a number of aspects, originating from both theory and ESSI guidelines and intentions forwarded to the PIE projects.

It is important that measurement data is collected and analysed properly. First prerequisite for that is a precise definition of the entity, attribute, and metric used. Although this issue has its origin in theory, it is not only an academic issue to be discussed between software metrics researchers. It is an essential issue for the practical applications. Without a well defined metrics, measurement data has no value to the organisation, and the measurement effort is a waste of resources. Even worse, the measurement data may be misleading as without precise definitions, all interpretations become possible. Without precise metrics, comparing measurement data from different organisations does not make sense either.

From this point of view, the PIEs studied appear weak. Naturally there is always bias in this kind of studies, and so may be even here: the reality might in some cases been better than it seems from the final report, especially as there is a lize limits for them. Inspite of that, it is surprising that EC has not required more precise objectives and results, expressed in measurable terms and measurement data. With such limited requirements on the PIEs, the added value of ESSI for the European market as a whole is considerably weak. It is not possible to determine whether the lessons learnt can be carried over to a candidate organisation using the same approach to improvement as there is clearly some ambiguity concerning how the measurements have been defined and metrics data collected and analysed.

To obtain an improvement in the software metrics maturity, software metrics and software process improvement should be included in the software engineering curricula, to train the top management and engineers of tomorrow to undertake improvement efforts and determine the magnitude of the improvement in a precise and comparable way.

References

[1] Caputo K: CMM Implementation Guide, Addison Wesley, 1998.

[2] EUREX – European Experience Exchange, Project Number 24478, ESSI Dissemination Action, Annex I, Project Programme.

[3] European Software Institute, http://www.esi.es.

[4] http://www.esi.es/VASIE/

[5] Fenton, N.E., Pfleeger, S.L., Software Metrics – A Rigorous & Practical Approach, International Thomsom Publishing Inc., 1996.

[6] Fenton, N. Whitty, R., Iizuka, Y., Software Quality – Assurance and Measurement. A Worldwide Perspective. International Thomson Computer Press, 1995.

[7] Fenton N: Software Metrics for SPI, Workshop on Process Improvement, Eurex, London, January 1999

[8] Grady, R.B., Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, 1992.

[9] Humphrey W: A Discipline for Software Engineering, Addison-Wesley Publishing Company 1995

[10] Humphrey W: A Discipline for Software Engineering, Addison-Wesley Publishing Company 1997

[11] Kan, S.H., Metrics and Models in Software Quality Engineering. Addison-Wesley, 1995.

[12] Kuvaja P, Bicego A: BOOTSTRAP - a European assessment methodology, Software Quality Journal, 3, 117-27, 1994.

[13] Orci T: Software Metrics in a European Perspective, EuroSPI98, Gothenburg, Sweden, 1998.

[14] Oskarsson, Ö., Glass, R.L., ISO9000 i programutveckling – att konstruera kvalitetsprodukter, Studentlitteratur, 1995.

[15] Paulk, M.C. et al, The Capability Maturity Model – Guidelines for Improving the Software Process, Addison-Wesley, 1995.

[16] SEI, Managing Software Development with Metrics, Course material, 1996.

[17] Shepperd, M. Foundations of Software Measurement. Prentice Hall, 1995.

[18] SPICE - ISO/IEC, Working Draft V1.00.

Session 11
SPI and
Measurement II

Chairman

Timo Varkoi

Pori School of Technology, Pori, Finland

SUPREME – a Statistical Approach to Support Project Estimation and Management

Rikke Sunde
Event AS, Oslo, Norway

Tor Vidvei
Event AS, Oslo, Norway

1. Introduction

In project planning, a core issue is to get the best estimates possible on duration, resource usage and overall costs of the project. The objective of the SUPREME project is to improve planning, estimation, monitoring and measurement of software projects by applying a statistical approach in project estimation and measuring.

2. Executive summary

Introduction

To improve the planning, estimation, monitoring and measurement of software projects, Event AS has integrated a project planning system with routines for collecting reliable data from the project activities and established routines for analysing the project data.

What is being achieved

SUPREME aims at integrating the collection functions for planning and accounting data, so that the actual development of the project in question can be compared to the project plan at any chosen time, and that the data and experiences gained can be utilised systematically to improve the planning process. Much of this may be achieved by informal methods; i.e. by collecting historical data and establishing appropriate descriptive reports. Some issues, however, call for statistical methods, especially the development and calibration of estimation models.

3. Business motivation

Introduction

The overall objective of the SUPREME project is to improve the planning, estimation, monitoring and measurement of software projects.

The motivation

Most small-scale project management tools lack effective mechanisms to support the learning from project experiences, in particular functions for collecting reliable data and to do statistical analysis directed toward the improvement of project planning and estimation. Project or man-hours accounting systems, on the other hand, often contain data that are distorted from a statistical point of view. This will probably be the case if e.g. the reported man-hours are the basis for calculating the customers' payments or subject to budget limitations, or if incentives or other “disturbing” considerations significantly influence the reported figures. Even if the project planning and accounting systems are linked together with data exchange etc, much relevant information about the project histories are lost - with poor learning as a consequence. This is the problem that is addressed in the SUPREME projects.

4. The project model

Introduction

A statistical method analysing long-term and short-term historical project data stored in a project database is applied to achieve the project goal. This method is supported by a configurable software tool, which allows continuos data collection and production of reports with status information and statistical results at any time during the project. Thus revised estimates for projects with highly uncertain and numerous milestones and deliverables will be produced at regular time intervals to achieve better control and management of projects. Emphasis is placed upon achieving results that are easy to use and interpret, rather than on sophisticated statistical methods.

The project model

The project plan is made up of two types of *entities,* each organized hierarchically, i.e. in a tree-like structure:

- *Activities*. The tasks of a project. The activities are aggregated in two separate hierarchies or *dimensions*, representing the "physical" organisation (program modules, functions etc), and the "accounting" organisation reflecting how the various activities are attributed to clients, contracts or subtasks within a contract.

- *Resources*. Actual persons, which are the only significant resource in our projects. The aggregation hierarchy reflects the organisational hierarchy, i.e. workgroups, departments etc.

When a project is planned, all its activities will be estimated with respect to resource usage as well as time schedules, before the project starts. This baseline- or reference-plan plays an important role both in the monitoring of the project and in the analysis after the completion of the project.

As a project is developing, resource usage, status changes and revisions of the estimates are registered daily or at certain other time intervals so that the current status and the latest forecasts can be displayed at any time. This registering is done for the elementary activities and resources, and automatically aggregated to higher levels. The history of all variables is saved, possibly with annotations, so that the entire history of the project may be analysed at later stages. All persons involved in a project are encouraged to add comments to their registrations, especially when estimates or status variables are changed.

Activities

A project is represented as a set of *activities* organised in a tree-like structure or *work breakdown structures* (WBS). "Activities" represent the various tasks in a project from a "physical" point of view. In the context of software development the activity- tree mirrors the organisation of the code in modules, classes, methods etc.

Various aggregation rules are defined for the variables describing an activity. The resource usage of an aggregate activity is calculated as the sum of the resource usage in all of its sub-activities, the starting date is found as the first starting date among the sub-activities etc.

To be registered as an activity, the task has to be of a certain size, at least one working day. Activities, which could be defined as rather small, will also be registered, but only containing actual closing dates and with no estimates for resource usage. This will typically be an activity lasting less than one day.

The activities are also organised in a tree-like structure representing the various external and internal projects from a financial point of view, reflecting how the resources spent on the activity will be accounted for. Aggregates within this hierarchy are called *accounts*. An account may represent a particular contract while the sub-accounts may represent various sub tasks within this contract. The accounts are used for budgeting and reporting to the clients. Together with the activity hierarchy, the accounting hierarchy represents a two-dimensional WBS of the activities. This is especially important within an object-oriented framework where code is written for reuse, possibly across various external projects. For projects that are to be accounted for on a per hour basis, the accounting will not be in accordance with the physical organisation of the activities.

Resources

In most software projects in Event AS the only significant resource type is persons or man-hours. Most machinery can be considered a fixed cost. Each employee is represented as a *resource* entity in the project model.

5. Data collection routines

Introduction

Data collection routines should be established to ensure that the collected data are of good quality from a statistical point of view. This means the ties between data collection and accounting, influence from budget considerations and incentive systems etc, must be minimised or at least brought down to an acceptable level. The reporting of resource usage, progress, estimates and forecasts should reflect the actual development and real expectations.

Routines

Data will be collected daily or periodically for activities, accounts and resources. The variables we collect data for fall into two categories:

- *Static*: Data that describes an activity or resource, and that are constant throughout their lifetime of the entity. Examples: Name and description of the entity, registration date,
- *Dynamic*: Data that describes the development and current state of an activity or resource. These variables may be updated continuously as the project evolves, or at certain intervals, when the project status and estimates are reassessed, e.g. at project meetings. Examples: Status, Estimated resource usage or completion date.

For the dynamic variables, the history of the variables will be saved, i.e. a pair of date and value each time the variable is changed. These historical data will be analysed periodically with certain statistical methods (event history analysis).

For each registration (data item), it is also possible to add annotations. Annotations should be added when they may be significant to the analysis of the project in later phases. (New variables can be added at later stages. If the annotations contain information that is needed in the statistical analysis, it is possible to create new variables and code the information.)

The variables may be aggregated following the tree-structure that the various activities or resources are organised within. The user controls the aggregation rules.

It is possible (and normal) to add new activities or resources during a project period. Such changes do not create any problem for the estimation and accounting routines in most cases. It is also possible to move and even delete an activity, but one should be aware of the following:

When an activity is moved from one place to another within a tree, its entire data history (even the data that are collected before the move) will follow and be aggregated within the new tree-structure, and consequently removed from the old tree-structure. The aggregates will display a history of its variables as if the activity has always been placed in its new structure.

When an activity is deleted, the user will have the opportunity to move its content to another activity. Despite the problems just mentioned this could be useful, e.g. in cases when a project plan appears to be too detailed as the project evolves. Then the activity tree can be “Cleaned" even after the project has started.

6. The methods

Introduction

Experience proves that a qualified and experienced person may be able to come up with good "guesstimates", which implies that there is some kind of recognition or “Repetitiveness" involved, although it may be very difficult to establish quantitative measures. To support this ability of reasonable "guesstimates", *feedback* is very important. If one never gets information about how reality compares to ones estimates, (or if the estimates become messed up by a lot of interfering changes in circumstances that are never accounted for), it will be very difficult to learn anything from experience.

Say, one is told that over a period of time one systematically underestimates projects with about 20%, and that most of these mistakes are caused by tasks added during the project period. On the other hand, one could at the same time hit fairly well with the estimates for programming and documentation efforts. One may be helped significantly in performing corrective actions and improving future estimates by this feedback, even without a formal estimation model.

The need for status reports and statistics with focus on the comparison of estimates made and results achieved is enforced by the common experience of most software projects: the early estimates tend to contain a significant amount of "guesstimates", usually more than in other types of projects e.g. within construction. This element of guessing is more prevalent the more “Creative" or the less routinised the project is. Estimation relies on quantitative measures for the outcome of the project or its activities at some level (e.g. code lines, modules, reports, pages of documentation), assuming some kind of repetitiveness.

COCOMO, Function Point Analysis and Work Breakdown Structure [3], [4], [5]**:**

The codelines or reports must be assumed to be of comparable degree of difficulty, implying that one must at some level be able to say "We have done something like this before". This is true both in the COCOMO and the WBS-based methods. But one of the main objectives of the advance of software development methods, not to mention software by itself, is to remove the need for repetitive or routine operations, so that most efforts may be directed toward creative tasks. One of the main perspectives of the introduction of object orientations is to provide e.g. mechanisms and language constructs that will ease the reuse of software, in order to avoid repetitive work. The various technologies for software integration (OLE, COM, CORBA etc) may be viewed from the same perspective. Most software projects, therefore, usually contain a significant element of creativity, which complicates the estimation and planning process.

It should also be emphasised that even quantitative estimation models like the COCOMO model relies heavily on judgement or qualified guessing. Although the expected costs in man hours are calculated from the relation between lines of code and the expected resource usage, which is the result of statistical research, one does not really know the number of codelines in advance when one is going to estimate a new project. The element of guessing is not eliminated from the estimation process; it is rather isolated to some parts of the estimation procedure, in this case to the assessment of how many codelines that is required to fulfil a certain task. After all this procedure might in many cases prove better than relying entirely on qualified guessing, probably because predicting the number of codelines is simpler than predicting the costs directly. But you may also have cases where the opposite is true, namely that it is better to assess the costs directly rather than first trying to assess the number of codelines, then to select an appropriate estimation model and finally calculate the costs.

Event history analysis [1], [2]**:**

In order to compare the estimates with actual resource usage, and thus give us a sort of feedback on the estimates, an approach based on *event history analysis* is used.
The method itself will usually be applied on historical data and not to perform the estimates (prediction) during the planning phase. This will make it possible to use the method to compare the estimates with actual recourse usage etc and thus give us a sort of feedback on the estimates.

This event history analysis is relevant while analysing event history data, i.e. data describing sequences of dated discrete events. A typical field of application for this method is demographical analysis, with the analysis of fertility and mortality depending on age and other factors as one of the core issues. Demographic data typically consists of individual "event histories" of birth, marriage, death, changes in health status etc. Another typical application of this method - which is also known as "failure time analysis" - is the analysis of the duration of products until a failure (or some other interesting event) occurs. The method focuses both on the analysis of *which events* or transitions that takes place, and on the *duration* between the various events, according to some timescale or any other variable that grows as time passes.

The "history" of a project can be described in similar terms as sequences of dated events associated with the individual activities within the project: registration, decision, start, completion, cancellation etc. We are interested in the transitions that takes place (what is the probability for an activity that has actually started, also to be completed?), and the duration according to various timescales and resource usage (how is actual resource usage and duration of activities distributed, compared to the original estimates?). We think that event history analysis is well suited to perform the analysis necessary to answer this type of questions.

In SUPREME we observe all events of interest for each activity, and aggregate them as individual event histories. A set of event histories is called event data. When aggregating the stories for a group of activities or a project, one will have a summery of the transition between several states. On the contrary the cross section data will only give information about how many activities that is in the state at any point of time.

An event history model is a schematic way of showing possible courses of events. The model used in SUPREME is containing 5 states: registered, approved, in progress, completed or rejected.

Suppose one is observing the events in a group of activities. The phenomenon that is of interest is “Completed”. The event history for two activities may be as follows:

Activity 1	
02.04.1999	Registered
15.05.1999	Approved
17.05.1999	In progress
15.10.1999	Completed

Activity 2	
15.04.1999	Registered
01.07.1999	Approved
17.08.1999	Rejected

Fig. RSU.1: Example of event histories containing 5 events.

By registering these events we will have an exact registration for the time each event occurs.

A set of states as described above are characterised as a case where there exist more than one transition from one state (competing risks of e.g. being “Completed” or “Rejected”). There are several entrances to some states (all activities are under risk of being “Rejected” at any point of time). In addition it is likely to assume that the intensities are depending on more than one time scale e.g. calendar time (date), time in state (for example time from “Approved” to “In progress”) or time in “Approved” and cumulative time in state (aggregated time an activity has been exposed to e.g. being rejected).

The possible states and transitions can be illustrated as follows:

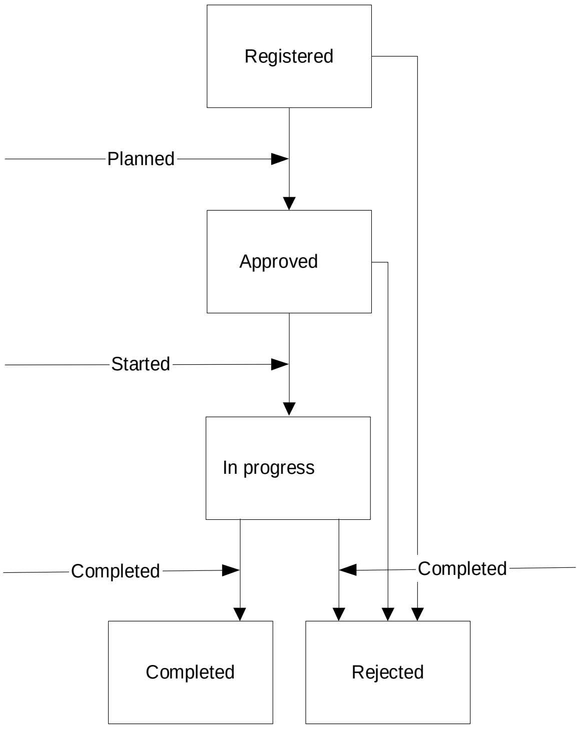


Fig. RSU.2: Transitions and states

The boxes represent the possible states each activity can be in. An activity can not be in all of them simultaneously, as an activity can not be "In progress" and "Completed" simultaneously. The transition between the states is marked with arrows.

To illustrate what is happening in the transition, we can regard a group of activities in a project. As time goes by, all activities will be completed or rejected. At the start none of the activities will be in the state "Completed" or "Rejected", but as times go by an ever-increasing number of activities in the project will be in the state "Completed" or "Rejected". But after some time, all activities will reach these final states. In our terminology all activity that enter the registered-state will be the same activities which are entering the completed or the rejected-state.

The transitions between the states can be illustrated as a *survivor function*. An example of this is shown in figure 4. For a given number of activities, the graph shows the estimated probability for an activity still being ongoing at each point of time.

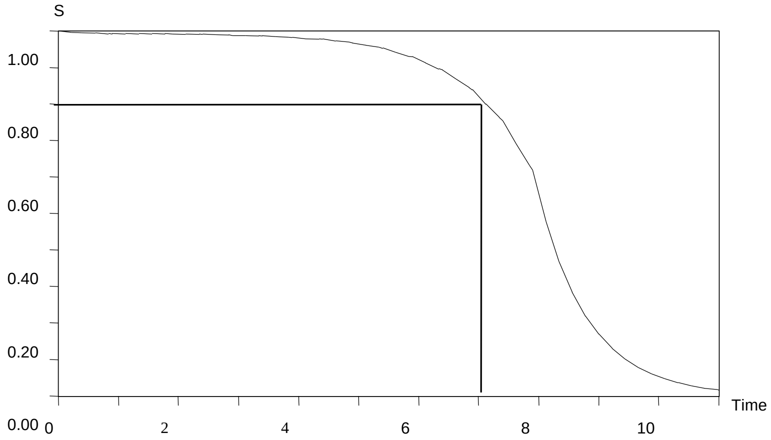


Fig. RSU.3: The survivor function

In this example an activity has a probability of 90% of still being ongoing after 7 months. After 10 months the probability of still being ongoing has fallen to 15 percent.

When an activity leaves for example the state “In progress”, it will either reach the state “Completed" or “Rejected", which make it necessary to define the intensities for each of these events. At any point of time the activity in progress is being exposed to the risk of getting completed and to the risk of being rejected. In the terminology of event history analysis, there are *competing risks* for a transition to either “Completed” or “Rejected”.

The use of event history analysis can be illustrated by one of the baseline projects - PETRA. We want to investigate how the resource usage in the project compares to our original estimates. In terms of the state diagram (fig RSU.2) we are going to investigate the transition from “In progress" to “Completed" or “Rejected". As the project is still running, we will use our last estimates for the resource usage and assume that all activities will be completed. (Obviously, when the project is completed, the final accounts on resource usage and status for each activity should be used instead of these revised estimates.) As we want to look at the actual resource usage compared to the estimates, we divide the resource-usage (actual or last estimate) by the original estimate – and use this as our "timescale". (See the last column in the table RSU.6) As the size of each activity varies, the data for each activity should also be weighted with the original estimates (taken as a measure of the "size" of each activity).

No		Activity	Orig estim		Rev estim		Rev/orig	
2.1		Planning	30	30	30	30	1,000	1,000
2.2		Documentation		330		475		1,439
	2.2.1	User doc Event	120		185		1,542	
	2.2.2	User doc Petra	90		120		1,333	
2.3		User assistance	120		170		1,417	
2.4		Debugging, improvement etc		640		650		1,016
	2.4.1	Debugging and error corr	90		160		1,778	
	2.4.2	Access contr & prot of data	60		65		1,083	
	2.4.3	GUI, various improvements	90		110		1,222	
	2.4.4	Multilevel undo functions	90		65		0,722	
	2.4.5	Improved error reporting	30		25		0,833	
	2.4.6	Functions for tracing changes	75		40		0,533	
	2.4.7	3D Spreadseet functions	45		30		0,667	
	2.4.8	Optization	120		120		1,000	
	2.4.9	..	40		35		0,875	
		Total	1 000	1 000	1 155	1 155	1,155	1,155

Table. RSU.4: The PETRA project

Weight	tE	Occ
30	1.000	1
120	1.542	1
90	1.333	1
120	1.417	1
90	1.778	1
60	1.083	1
90	1.222	1
90	0.722	1
30	0.833	1
75	0.533	1
45	0.667	1
120	1.000	1
40	0.875	1

Table RSU.5: The Event history data for the transition from
“In progress" to “Completed"

In the RSU.5 the “Weight” column is containing the original estimates for each activity. The tE column shows the resource-usage relative to the original estimates. In the Occ column a 1 or 0 indicates that the activity was- or was not completed.
The event history data derived from this is displayed in table RSU.5. Data for the

aggregates are excludes in order to avoid counting the same activity multiple times. A hazard function and an associated "survivor function" expressing the probability distribution for the resource usage relative to the estimates, can be estimated from these data. The survivor function is displayed in figure RSU.6:

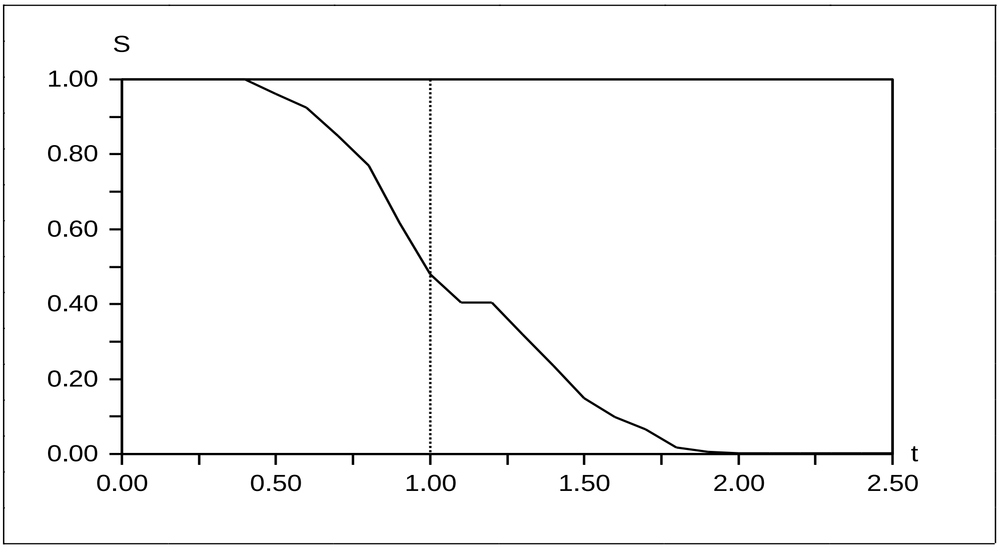


Fig RSU.6: Survivor function for the transition from “In progress" to “Completed"

The figure indicates that the project is somewhat underestimated (the survivor curve is "skewed" to the right in the figure - the area above the survivor curve to the left of t=1 is less that he area below the curve to the right of t=1. (The average resource usage is 15% higher than the estimates.) Further, the resource usage compared to estimates varies between 0.40 and 1.75.

This type of figure gives in a glance a summary of the performance of the project. The figure RSU.7 contains some other stylised examples of "survivor functions".

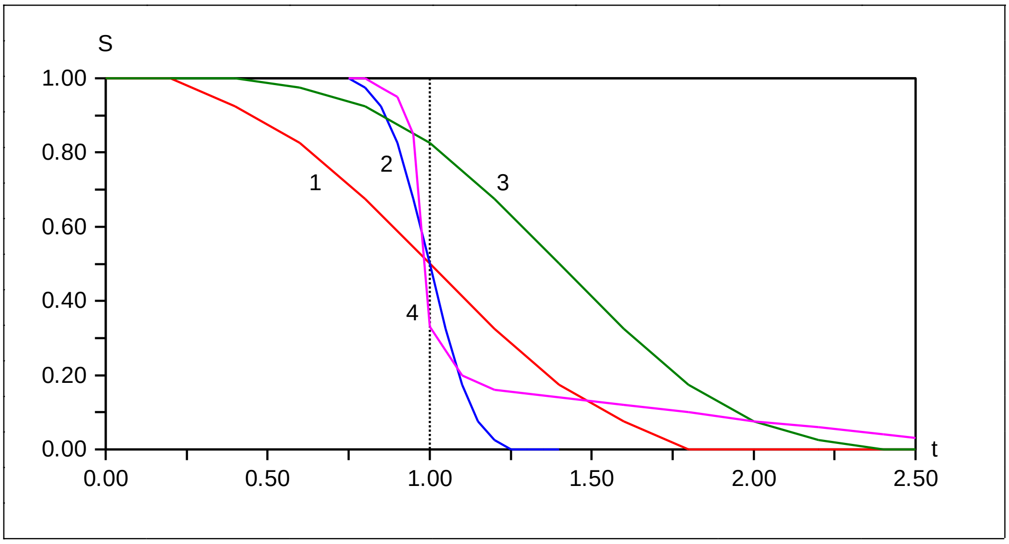


Fig RSU.7 Examples of survivor functions

Graph 1 and 2 both indicates that estimates are unbiased, as the average time to complete an activity is equal to the estimated time. But graph 2 indicates less uncertainty: the curve is closer to the t=1 line on both sides. Graph 3 indicates that estimates are biased: for most activities and on average the time to completion is longer than estimated in advance. Graph 4 also indicates that the average time to completion is longer that the estimates, but in this case most activities are completed within the estimated time.

As mentioned above each activity that is “In progress” will be under risk of being “Rejected” or “Completed” at any point of time. On this basis one can estimate a hazard rate for each of these risks to illustrate the probability of an activity being “Rejected” vs. “Completed”.

One can obviously assume that an activity’s progress depends on several explanatory variables. This could be e.g. the activity-size and -type. Being aware of this may help to identify critical activity types or other states, which are sources of uncertainty (or risk).

Meanwhile one has to consider what will happen if one is facing major deviations from the original schedule. When an activity is running out of time, does that means that one will use all available (and not available) resources on that particularly activity, forgetting the other activities. And what if an activity is doing more than just fine, so it will be completed “too early”? Will one still use all the planned resources just to get oneself a well-deserved breathing space?

7. The expected impact.

A well designed framework for the estimation of software projects will have potential positive impacts on the business in several ways:

- Timing of delivery can be done more precisely, with less uncertainty.
- Potential bottlenecks, in time and volume; e.g. problems which can only be resolved by a particular person, will be identified more easily.
- Diversions from plans will be explained to customers more easily.
- The management of the company will be simplified because corrective actions can be undertaken at an early stage of a problem.
- More effective resource allocation and usage will be possible, especially when it comes to long-term development of the software.
- Communication between all members of the development team will improve.
- Development time will be saved through well-documented costs and estimates when negotiating contracts.

8. References

[1] T.Bragstad, M.Nielsen, T.Vidvei, J.Østervold: Forløpsmodellen MOTIPE, presentasjon og analyse (the event history model MOTIPE, presentation and analyses), The National Insurance Administration, 9/94.

[2] H-P. Blossfeld, G. Rohwer: Techniques of Event History Modeling, LEA, Mahwah, New Jersey, 1995.

[3] J. E. Matson, B. E. Barrett, J. M. Mellichamp: Software Development Cost Estimation Using Function Points, IEEE transactions on software engineering, Vol. 20 no 4, April 1994.

[4] C. A. Behrens: Measuring the productivity of computer systems development activities with function points, IEEE transactions on software engineering, Vol. SE-9, no 6, November 1983.

[5] Breien, Andersen, Jonassen, Stålhane, Urstad: Estimering- og fremdriftsmålemetoder for utvilkingsprosjekter, ITUF R-29, 1992.

Appendix 1: COCOMO and Function Point Analysis [3], [4], [5]:

The COCOMO method is based on inputs relating to the size of the resulting systems and a number of “Cost drivers” that affect the productivity. The cost function in COCOMO can be written:

Cost = a • LOC^{b} • ∏*f*$_{i}$,

where “Cost” is the total project cost, ”a” and ”b” are constants and dependent on the development modus only, ”LOC” is the size of system, measured by Lines Of Code and ” f_i” are the cost drivers. The COCOMO model does estimate the cost of production from the system size (measured by lines of code (LOC)), the type of development modus (organic, semidetached and embedded) and the cost drivers. Examples of cost drivers that affect the productivity are product attributes (data base size), computer attributes (execution time constraints) and personnel attributes (programmer capability)

Unfortunately the COCOMO model does not provide an opportunity to balance the interests of total cost and the time of realisation in a project. The uncertainty of the estimate will be enlarged as a result of implementing the cost drivers.) ….til hit!

As method to be used to make proper estimates the function point analysis seems to be the most relevant. This method which is quantifying the size and complexity of a software system in terms of the functions that the system delivers to the users, is unrelated to the language or tools used to develop a software project. As for projects, which are based on object orientations, this is of great importance

To measure productivity, a product and a cost is defined and measured. The product is the function value delivered to a user. The number of inputs, inquires, outputs, master files and interfaces delivered are counted, weighted, summed and adjusted for complexity. The collection of function point data has two primary motivations. One is the desire to monitor levels of productivity, for example number of function points achieved per work hour expended. Another use is in the estimation of software development cost.

Appendix 2: Presentation of the authors

Rikke Sunde: Cand Polit (Master of Science degree in economy), born 1969

Education:
- Economics
- Demography
- Business administration

Experience:
- Event AS, from September 1998
- Ministry of Finance, 1997-1998.
- NORAD (Norwegian Agency for Development Cooperation), 1995-1997.

Reports:
- "An empirical analyses of the two-sex problem – to be used in projecting population sizes", 1995.
- "The use of Norwegian resources in the Norwegian development aid 1994"

Tor Vidvei: Cand oecon (profession oriented degree in economics), born 1955

Education:
- Economics
- Informatics

Experience:
- Event AS (leading manager) from 1992
- Norwegian Computing Center1990 - 92
- Gruppen for Ressurstudier (NTNF) 1989-1990
- University of Oslo, Department of Economics 1983 - 1989

Reports:
- H. Brunborg, T. Vidvei: "Parity Specific Population Projections Using Stochastic Micro Simulation". (Scandinavian Population Studies, p 273-91, Oslo, 1989
- T. Bragstad, M. Nielsen, T. Vidvei, J.Østervold: "Forløpsmodellen MOTIPE, presentasjon og analyse (the event history model MOTIPE, presentation and analyses). (Rapport 9/94 The National Insurance Administration)

Appendix 2: Presentation of the company

Event AS is a small enterprise developing software systems for statistical modelling. The applications are tailor made for each customer, while the development process relies heavily on the use of a software library called EVENT, which is being developed continuously as part of each of the application projects. This approach communicates the management of the development processes in many ways, especially due to the ties and dependencies between the different application projects that are crated through the use of the software library.
The company presently has three employees.

Among our major projects are:

- EVENT: An event history modelling program (for statistical analyses and prognoses based on event history analyses). The program forms the software library do be used in all other projects.

- TRYGD: A forecasting model for the Norwegian social security system
  (Client: The National Insurance Administration)

- PETRA: A short-term forecasting model for the Norwegian oil activity
  (Client: The Ministry of Petroleum and Energy)

- PROGNOSE: A forecasting model for pension arrangements in companies and institutions.
  (Client: DnB/Vital - A major Norwegian Bank and insurance company).

- PROMAN: A project management tool (Only used internally so far. May become a product for sale later.)

Introducing professional project management in an SME

Carlo Giordani
Cortis Lentini s.p.a.
Via Daste e Spalenga 16
24020 Gorle (Bergamo) – Italy
fax: + 39 035 304611
e-mail: carlo_giordani@cortislentini.it
Esprit project n. 27370 PRO^3

Abstract

This paper presents the initial results, a few moths before conclusion, of the PRO^3 Process Improvement Experiment, ESSI project n. 27370.

Projects are difficult to manage because project plans are not precise. They are not precise because the estimation of size and duration is empirical, and because detail in planned activities is not sufficient. As plans are not precise, project tracking is difficult to make, and cost/effort monitoring is made too late.

The goal of PRO^3 is to solve these problems by:

- introducing the use of Function Points to estimate size and duration of project
- introducing project planning tools and methods (Gantt charts, Pert charts, deliverables and milestones), and a decomposition of the project into finer grained activities to improve project planning
- introducing project tracking tools and methods to regularly compare planned and current state of the project

Introduction

In the following we describe the context of the PRO^3 PIE: the company in which it is performed, the software process before the experiment, and the phases planned for the experiment.

The company

PRO^3 is proposed by Cortis Lentini (shortly C/L), a SME incorporated in Bergamo, Italy, in 1978, offering standard software products for information systems, and their customisation to specific needs. The products, used as basis for a large number of projects, are: **UNISMART**, general purpose information system, mainly used for administration and management; **SICO**, strategical analysis system; **OPENPROD**, object oriented platform for the development of production management systems; **GOLD**, suite of CAD applications that improves the design productivity; **SAXTECH**, industrial automation control system.

Furthermore Cortis Lentini offers consulting on: organization and management; object oriented analysis; information systems design and implementation; development of ad hoc interfaces between existing I.S. and new solutions; tuning of existing systems.

The numbers of Cortis Lentini: 6 productive centers, about 150 employees and a turn-over of 13 million euro in 1996.

Since 1978 Cortis Lentini main mission was customer satisfaction: a strict quality policy was rewarded in 1994 with the ISO 9001 certificate.

Given these high level goals, the company identifies the project management area as essential to achieve them:

- customer satisfaction is increased if projects are delivered on time, at estimated cost and quality level
- profitability is controlled and possibly increased if the ability of forecasting precisely effort and duration of projects is increased
- competitiveness is increased if managers can, for each project, control and decide the level of profit
- the above is possible if suitable techniques are used for project management, in particular project estimation and tracking, and measurement of quality of the product/process
- the same holds if larger and more complex projects are to be managed.

The software process

All projects have a project manager (PM) who is responsible for the entire project management (and success) and provides feed-back between the technicians and the customer. All projects (both development of products and ad hoc customisations for clients) are based on a classic waterfall life cycle with a lot of feed-back points. This is mainly due to the new customer oriented approach: each customer has a particular point of view and deserves its specific information system.

So after collecting the user requirements, the project manager (PM) writes the requirements specification document, that is submitted for approval to the customer. If the document is approved, the PM, with the aid of a system architect if necessary, builds a functional definition document, a system description with emphasis on the system main components and the various software interfaces and hardware devices.

The next step is the design document, a collection of information necessary to the programmers to adapt the information system from the standard product, or to build a new one.

This step is vital for the project, so the PM needs to validate the solution with the customer because at this point the information system is detailed (on the paper) at the maximum possible level.

Then the design document is internally validated and the PM prepares the quality and development planning (PQS) and the verification planning (PDV): PQS details all development phases, PDV only the test phases and is completed with checklists. These documents are submitted to the customer for approval.

Now we can build the system and check with the customer each subsystem as it is completed. When the software passes all the test phases, it is released to the customer for the ultimate approval.

Quality Assurance

In addition to the PM , Quality Assurance roles are appointed at the beginning of each project:

- the Test Responsible, that executes the tests planned in PDV and listed in CHL (checklists);
- the Validity Responsible, a person chosen from the customer staff that approves the various phases of the project;
- the Quality Manager, a technician from Cortis Lentini that controls the correspondence between the documentation and the product, and warranties that the project is correctly, completely and well described.

Tools

Usually all the systems are built with the standard tools in use in C/L.

As hardware platform we use Hewlett Packard machines, both for servers and for clients: Cortis Lentini is the major italian VAR of HP for servers.

The operating system used on the the servers is HP*UX on the PA RISC based machines and MS NT on the Intel based machines.

The preferred RDBMS platform is Oracle, a very robust and scalable engine that we can apply from a little workgroup server to a large enterprise server.

The applications are built (and run) on the client site using several tools, according to the customer necessity and project tipology: for small new applications MS Visual Basic; for more complicated applications Uniface is the preferred tool; also we have built objects library for Gupta and Centura, and for particular projects we have used Delphi.

Evolution of the Quality System

The software process described above is the result of the definition of the quality system and its ISO9001 certification (a 42 person month project started in 1994 with 18 months duration), and of a continuous update of the quality system (4 person involved with an effort of 12 person months per year).

The 1994 version of the quality system was adapted for large projects, but too cumbersome for small projects. Since often projects in C/L are small ones, soon after certification the quality system was revised.

The second release of the quality system divided projects in two classes, **micro** projects and **macro** projects, but this solution was not so useful as we expected to be.

The third (and current) release of the quality systems returned to a single class of projects, but introduced a lot of optional documents that the quality manager can decide to use or not, in function of the size of the project.

The PIE phases

The PIE has the following main phases:

1. Preparation. An assessment of the current process is made to understand, both qualitatively and quantitatively, the current state. Measures are taken a posteriori on past projects for comparison with the results from the baseline project. Project management processes (estimation, planning, tracking) are designed, starting from state of the practice techniques and adapting them to the context of C/L. Then they are integrated into the quality system and quality procedures are updated. In parallel supporting tools are chosen and purchased. The measurement system (procedures, database for measures, tools and methods to collect data and compute measures) is set up.

2. Training. The staff working on the baseline project are briefed on the motivations of the PIE, on the main design decisions, and are trained on the techniques and tools introduced.

3. Experimentation. The new software process is experimented on the baseline project. Given the nature of the processes introduced, all phases of the baseline project are involved. Modifications to the process will be compared with similar past projects to understand if the weaknesses have been solved.

Project management

Before the PIE

We describe here the project management practices as performed in Cortis Lentini before the PIE.

Estimation of effort and duration for a project is made using the best judgement of the project manager in charge of it. Of course this method does not easily allow managers to resist pressures for faster delivery from clients and especially marketing staff.

Scheduling is made on the high level phases introduced in a previous section. In practice, only the delivery date for the client is recorded and monitored.

Tracking is informal, basically the issue of a document prescribed by the quality system (requirements, functional specification, etc) is used to track the state of a project. However, this approach becomes weak when coding and testing start. Given the number of functions and modules in a project, and the lack of tools, tracking their status is very difficult for the project manager.

A tool records effort spent per person per project. Effort data is not used by managers, but for accounting to clients. Sometimes effort data is used to verify if a project is profitable or not.

In short, project management before the PIE is made mostly informally, with little support from tools and procedures. Estimation, scheduling, tracking and post mortem analysis are not identified precisely nor clearly described in the quality manual. While this approach is still reasonable for small projects, it is definitely not suitable for medium and large projects. Project managers are aware of this problem and ask for improved project management practices in Cortis Lentini.

After the PIE

We describe here how project management activities have been changed by the PIE.

Overall, changes were guided by [3,4].

We describe them in terms of project management activities, project phases defined, tools used. Measures are used extensively as a quantitative base for project management, so they will be treated under all the following sections. A final section summarizes them and presents measures from past projects.

Project management activities

Project management consists of the following activities:

- Estimation: the size of the application to be built is estimated. The Function points measure [1,2], and the Function Points Analysis method are used here.
- Planning: using the number of function points estimated in the previous phase, considering the phases prescribed by the quality system, and past productivity figures, a schedule (in the form of a Gantt chart) is produced. The schedule defines phases, milestones and deliverables.
- Effort tracking: project members log the time spent per project and per phase. This allows to track the effort spent per phase, and to compare it with the plan. The completion of milestones and deliverables is tracked too. Measures such as estimation accuracy and earned value analysis are computed.
- Defect tracking: project members, and clients, log defects found on the product. Each defect is tracked until fixed. Measures such as delay to fix a failure, effort to fix a failure, failure density, failures found before and after product release are computed.

It should be noted that the activities are listed in a logical order. Initially, estimation is accomplished before planning, which comes before tracking and measurement. As the project proceeds, tracking is done continuously, while estimation and planning can be repeated if needed.

Project phases and project types

Estimation, planning and effort tracking require a clear definition of the phases in which a project is organized. Also defect tracking uses phases, for instance to log in which phase a defect was introduced and removed.

The phases defined are listed in the table below. Projects are classified in small and large. In function of their size some phases are dropped.

	Small to medium project	Medium to large project
Requirements	Yes	Yes
Functional analysis		Yes
Size estimation and planning	Yes	Yes
Design	Yes	Yes
Detailed estimation, replanning		Yes
Coding, unit testing	Yes	Yes
Integration, integration		Yes

testing		
Documentation	Yes	Yes

Figure 1 - phases and project types

The project management toolkit

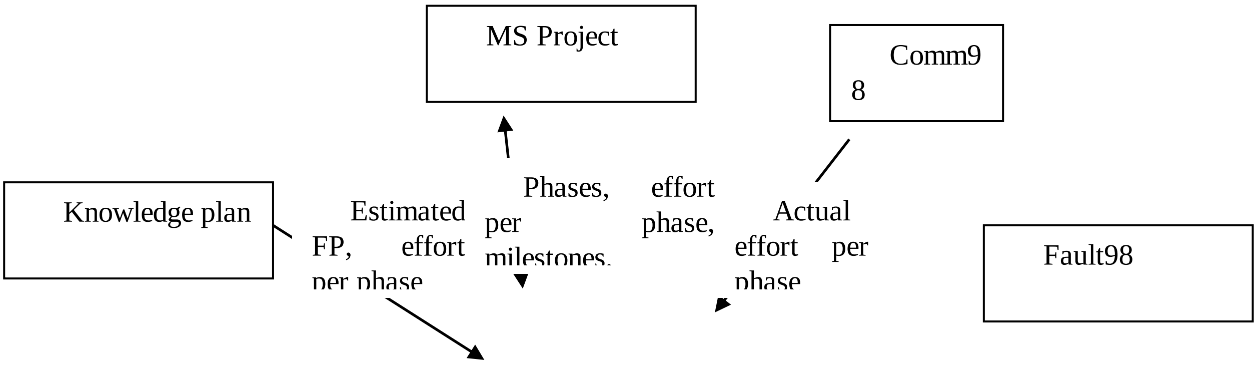
Project management can be considered an overhead, if it is not suitably supported. We have assembled a project management toolkit to automate its essential functions. In general the approach is to use existing tools, with special care to allow their integration. Integration has been accomplished by making them communicate only through the measurement database.

The measurement database is designed to be company wide, and to collect data for all projects. Also, it is assumed that tools can change, while the measurement database is a company asset that has to last. This is another reason why tools communicate only through the measurement database.

The tools selected are:

- Knowledge Plan. This tool automates estimation activities, basically it produces a function point estimate starting from a description of project characteristics and functionalities.
- MS project. It supports the planning activities.
- Comm98. This tool is an evolution of a tool built inside Cortis Lentini to describe projects and log effort spent on them. Each project staff member has a copy of the tool to log his effort.
- Fault98. Also this tool was built inside Cortis Lentini to track defects, and was partially modified.
- QARun. This tool supports testing activities, and provides testing coverage measures.

A standard spreadsheet is then used for analysis of measures starting from the measurement database.


MS Project
Comm9 8
Knowledge plan
Estimated FP, effort per phase
Phases, effort per phase, milestones.
Actual effort per phase
Fault98

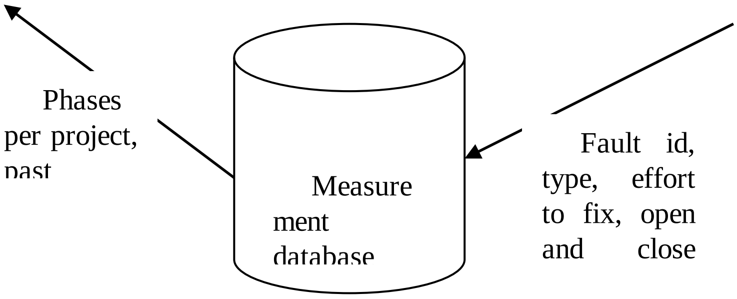


Figure 1 - tools and data exchanged among them.

A typical usage scenario is the following.

1. The project manager describes the project in Knowledge Plan. The tool, using past productivity figures, the phases to be used for that type of project, computes an FP estimate, and the effort per phase.

2. MS project reads from the database phases and effort per phases. The project manager defines temporal relations between phases, and knowing other possible constraints, defines a Gantt for the project, including milestones and deliverables. The Gantt is stored into the database.

3. As the project continues, the staff records the effort spent per phase using Comm98, milestones achieved and deliverables produced.

4. MS project reads this information and displays it on the Gantt. The project manager can visually understand the current state of the project, and replan if needed.

Project management measures

Measures used for management have been informally introduced in the previous sections. We summarize them here. The two main entities considered are project and defect.

Each project is characterized by.

- Development type (ex novo, customization, evolutive maintenance)
- Process type (small to medium, medium to large)
- Size type (small <=50 Function Points, medium (50 to 300FP, large > 300 FP)
- Estimated Function Points
- Actual Function Points
- Estimated Duration
- Actual Duration
- Total estimated effort
- Total actual effort
- Actual effort per phase

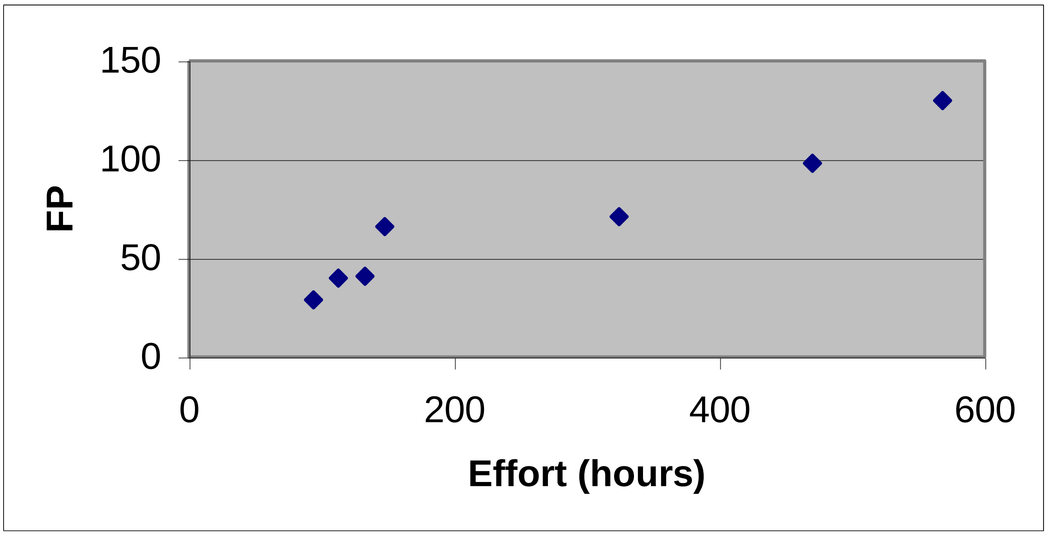
The thresholds for size type have been defined by clustering past projects (this issue will be discussed later). The process types are only two, and not three, as many as the size types, because at the current level of granularity for processes, no meaningful process for medium projects can be defined. Therefore, medium projects are allocated to the small or large process, using the judgement of the project manager.

Each defect is characterized by

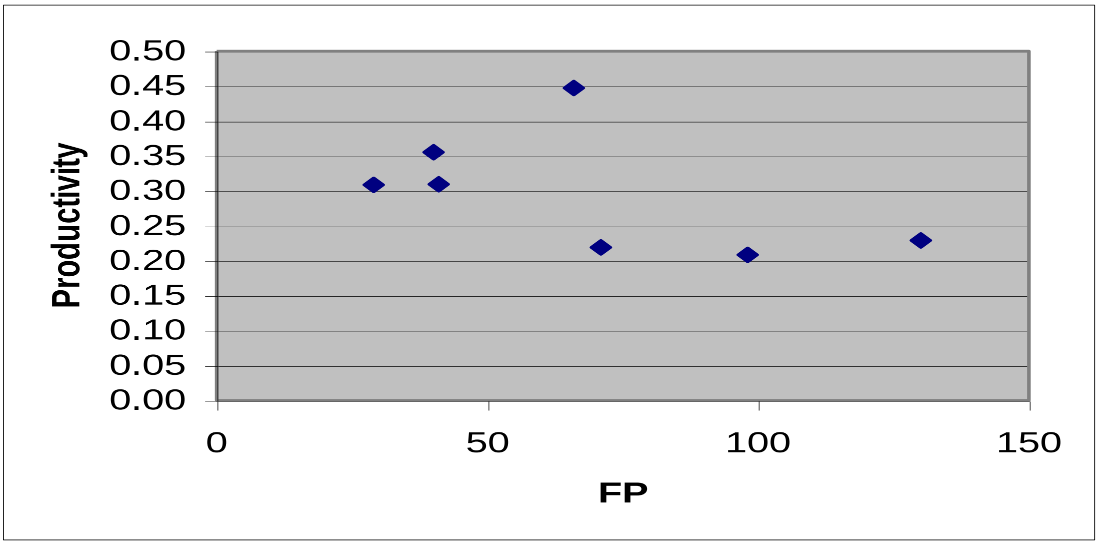
- Date when found
- Date when fixing started
- Date when fixing finished
- Effort to fix

A number of projects closed have been measured a posteriori, using data collected and interviews to compute or estimate the measures. The goal is to define a baseline of measures, both for estimation purposes, and for internal benchmarking. We present here some results.

The chart below presents a scatterplot Function Points vs. effort for the whole set of projects assessed. Most projects are small, a large project can be considered as an outlier and has been removed from the chart.



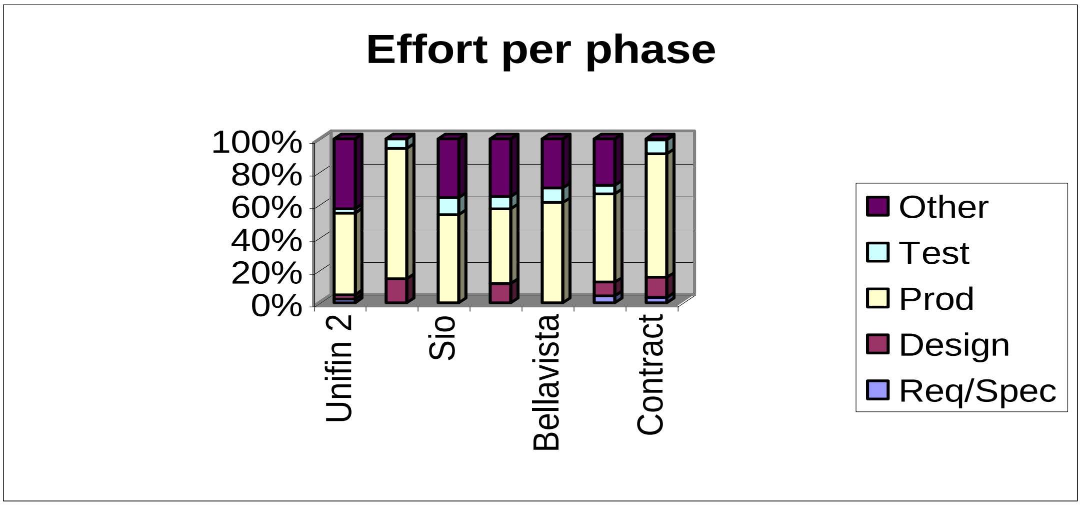
Two clusters appear. Small projects (less than 200 hours) and medium projects (300 to 600). Although very few data points are available, two linear relationships can be envisaged, for small projects and medium projects. As expected, the productivity for small projects is higher than for medium ones. Other attributes of projects (ex-novo vs customization vs evolutive maintenance) does not appear to have influence. On the other hand, small projects are always performed by one person, while medium projects involve at least two person teams, what could explain the productivity change. Only one project is available for the large category, so we cannot state anything on them.



Finally, we plot productivity vs. size. Now the clusters already observed in the plots effort vs size appear in great clarity. The small projects (1-50 FP) have productivity around 0.30. Then medium projects (50 to 150 FP) decrease productivity by one third to around 0.20. Finally the only large project (600+ FP, not in the chart) has a further drop in productivity. We exclude from this analysis as an outlier the project with productivity 0.45. We can also observe that the variation in productivity for small projects is much larger than for medium projects. This could depend on varying performance of individuals, and on their greater effect on single person teams than on larger teams.

Although few data points are available, these numbers confirm intuitive observations from project managers and suggest that both management practices and improvement actions have to be targeted in function of the size of projects.

The next chart reports the distribution of effort per phase. This information is used to calibrate the allocation of effort per phase during the scheduling phase. Moreover, some investigations have been started, to understand why the effort allocated for requirement specification is so low in nearly all projects, why design effort is sometimes absent, what the ‘other’ category actually contains when it is above 20%, why the other category is completely absent in some projects.



Conclusion

Work performed

Project management activities have been clearly identified and described. Measures are used as a support for all of them. As a starting point, a small set of measures has been defined and collected a posteriori from past projects to build a measurement baseline.

The main management activities defined are estimation, scheduling, tracking and post mortem analysis. Projects are classified in small and large, small projects perform a limited number of development and quality assurance activities. Clearly this has an impact on scheduling and tracking.

An integrated toolkit has been selected to support management activities. It is composed of tools to support estimation of size and effort, scheduling and tracking of activities, effort spent on and faults found on projects. A relational database acts as a repository of data from projects at the company level. A spreadsheet is used to compute and analyze measures.

Lessons learnt

The PRO^3 project is now a few months before conclusion, and experimentation of the project management infrastructure is on going. Although it is too early to derive the final conclusions on the effectiveness of the new management infrastructure, a number of observations can already be listed.

- Measures change the point of view on what it is analyzed, offer more insight and oblige to more precision. This is well known, but remains abstract theory especially in software. However, during the project a couple of cases convinced everybody that the theory was applying in concrete to our particular case. For instance the size of projects was traditionally measured by effort. But the analysis on the charts presented in this paper (two projects with same FP but 100% variation in effort) demonstrated that effort was misleading as an indicator, and FP should be used instead.

- The definition of phases in function of types of projects is delicate. The current definition in PRO^3 is the third iteration if we consider the whole history of the Cortis Lentini quality system. The availability of a tool infrastructure can change the context, allowing some more overhead for smaller projects. We have to wait for the completion of the experimentation to evaluate fully the effectiveness of our choices.
- Project staff is collaborating positively, enthusiastically in some cases. Initially we expected some resistance or skepticism. On the other hand the growing size of projects, and the related growing impact of problems, has made everybody aware that sound project management is essential to survive. Also, PRO^3 can count on solid backing from top management, and tries to involve Cortis Lentini staff with continuous communication about the project. Again, measures help a lot to focus discussion and ideas: managers were in first line in finding and discussing hypothesis to explain trends and outliers in analysis of measures.
- The integration of tools is sometimes tricky and requires a lot of effort to understand the intricacies of each tool vendor. Sometimes we were not able to obtain the exchange of information requested, and this resulted in missing the information, or requesting it more than once to the user.
- Some tools require a considerable effort to be learnt and used effectively. Initially we were planning to let each project manager use each tool. Now we have restricted the use of the estimation tool to a couple of ‘experts’ that can use them effectively.

Acknowledgements

This work was supported by the European Commission, under contract Esprit/ESSI PRO^3 n. 27370.

References

[1] Albrecht, A.J. and Gaffney, J.E. 1983. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Trans.* 6: 639-648.

[2] Jones, C. *Applied Software Measurement*. Mc Graw Hill, 1996.

[3] Fenton, N.E., Pfleeger, S.L., *Software Metrics – A Rigorous and Practical Approach*, International Thomson Press, Boston, 1997.

[4] Grady R., *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, 1992.

Experience from process improvement in a SME

Hans Jørgen Lied, M.Sc.
Telenor Geomatikk AS, Trondheim, Norway

Tor Stålhane, Ph.D.
SINTEF, Trondheim, Norway

Introduction

This paper describes a case study that was carried out in 1997-1998 and supported by the Norwegian national research program SPIQ. The company involved wanted to find a way to identify problem areas in the software development and to reduce the amount of rework caused by the introduction of errors during software development. The case study was motivated by the challenge of finding a way to identify problem areas by introducing the process improvement framework offered by SPIQ.

Our opinion is that the best way for a company to improve is through learning from their own data and experiences. In order to perform data analyses, we therefore have to combine collected data with experiences that are available in the organisation. We need to decide what to measure and how, how to make sense of the data and how to use the knowledge. From SPIQ we got the idea to use a combination of three methods:

- Goal Question Metrics (GQM), to decide on what and how to measure.
- Pareto analyses, combined with a robustness analysis to analyse the collected data and to assess the confidence we should have in them.
- Root Cause Analyses (RCA) [8], augmented by use of the Ishikawa diagram [7], to identify the root causes of our problems.

The results from these analyses identified the main problem areas. The feedback sessions in the GQM method were used to combine data, prior knowledge and experience during data analysis. Our approach and results will be useful for other organisations facing a similar challenge.

The collected data open for several interesting analyses such as how the differences in the projects influenced the types of errors, their corrections cost and time to discovery. We will in this paper, however, only focus on how to improve the development process so that we get fewer errors in the future. The rest will have to wait.

The rest of this paper is organised as follows. The first section describes the SPIQ program. The next section describes the organisation and projects on which this case study was performed. After this we describe the approach we chose. Next we describe the analyses we did and the results we obtained. Finally we present some conclusions.

Software Process Improvement for better Quality (SPIQ)

The SPIQ program was started in April 1997 and is sponsored by the Norwegian Research Council (NFR). Its main goal is to "increase the competitiveness and profitability of Norwegian IT-industry through systematic and continuous process improvement". The SPIQ program is based on the process improvement principles of Total Quality Management, [1], [3], GQM and the experience factory [5]. See http://www.geomatikk.no/spiq for more information about SPIQ. The process improvement offered by SPIQ builds on three pillars:

- The PDCA – Plan, Do, Check, Act – cycle.
- All decisions should be based on facts – observations and experience
- Developer participation is an essential part of all process improvement activities.

The work described in this paper has benefited from SPIQ in several ways:

- SPIQ has provided valuable method support
- SPIQ has motivated and partly financed the internal work of introducing process improvement methods to the company
- The methods, experience and changes were discussed at the SPIQ meetings

Environment

The company where we did this case study is a medium sized Norwegian company. It has a total of 270 employees with offices in Trondheim, Oslo, Tønsberg and Kristiansund. In addition they have offices in Sweden. At present the software development division employs 14 persons.

The software development division is of a size that makes it possible to easily adapt to changing market demands. At present the market wants tailor-made systems and special changes performed on already existing software products. There is a large amount of companies that deliver solutions in the same area and the competition is fierce.

Projects are often undertaken, based on a fixed price, which demands a streamlined development process and little room for rework caused by the introduction of errors. Systems with few or no errors are also important as a way of selling stability and reassurance to customers who invest money into software development. A happy customer is likely to return.

The two development projects were chosen because they were quite different and, at the same time, representative for projects this company is involved in. The only parts that were identical in the two projects was the project team and that they used the same development process model. The differences are much more prominent. The most important ones are:

- Different methods of development; Project 1 had its structure given right from the start, while Project 2 depended on prototyping.
- The customer defined quality assurance requirements for Project 1.
- Project 1 started with a readily made requirements specification from the customer, while Project 2 started with an idea from the customer and the requirements specification was a co-production between the customer and the developers.

The role of GQM in Root Cause Analysis

Why we chose the GQM method

Our first problem when starting to use RCA was the lack of a method for collecting data to be used. There was no direct support for this in the SPIQ framework but we found a way of combining parts of GQM – mainly the GQM abstraction sheet - with the Pareto analyses that proved to be efficient. In addition to the abstraction sheet, another important reason for using GQM was its strong focus on developer participation. We dropped the lower part of the GQM abstraction sheet – the part that contains the hypotheses concerning values of the focus parameters and the hypotheses concerning the impact the variation factors had on the focus parameters. To fill in these two quadrants would have been interesting, but was dropped since the data later should be used in an RCA setting.

The GQM abstraction sheet was used to manage and structure the brainstorming sessions and to identify the causes we would like to analyse in the later RCA. By getting a strong participation, we hoped to obtain two things:

- More reliable data. The developers should presumably be more interested in obtaining correct data if they felt that they had ownership to the goal and purpose of the data collection.
- More interest in the results. One of the factors that can really kill an SPI initiative is lack of interest from the participants. By involving the developers from day one, we hoped to create this interest and thus keep the improvement program alive.

In addition, we would get a better set of basic root causes to analyse if we tapped the large amount of experience that is available from the developers. Without this knowledge, we would have been forced to collect data on many more possible causes, thus creating a large and possibly unwieldy model. This would have cost extra resources without adding anything to the value of the final results.

The GQM process

Since none of the project participants had any prior knowledge of GQM, we started with a two hours GQM workshop. This was done with the help of research scientists from the SPIQ program. As in earlier cases, it turned out that GQM was easy to understand and use. During the two hours session we were able to describe the method and work through the goal that the company had defined. We use this as a standard approach, as opposed to start with a toy example. One of the strong points with GQM is that it tries to maximise the use of expert knowledge. A toy example will thus not help the users to understand why GQM is a good idea.

In order to get a practical set of failure causes categories, we started with the following GQM goal: **Analyse** *the reported failures* **in order to** *understand causes for failures* **seen from** *the developers in the X project*. The GQM process produced a GQM abstraction sheet, which was converted to a data collection form. The questions were converted to entries in the form. This process was straightforward and gave a practical data collection form. The data form represents the metrics in GQM.

After the initial process only small adjustments were needed. This was done in the first feedback session, where component complexity was added as a failure cause. The collection form had the following categories for reporting a failure:

Focus – main cause

1. Incorrect use of cut-and-paste from old code
2. Incorrect reuse of old code
3. Incorrect use of language features
4. Incorrect use of library components
5. Incorrect attempts to make the code general - prepare it for later reuse
6. Attempted reuse by introducing global variables
7. High component complexity
8. Large, unanticipated variation in input data

9. Incomplete or bad test data
10. Too large code components
11. Missing or incomplete configuration management
12. Incorrect component integration
13. Insufficiently detailed requirements specification
14. Wrong code logic
15. Errors in hardware – software communication

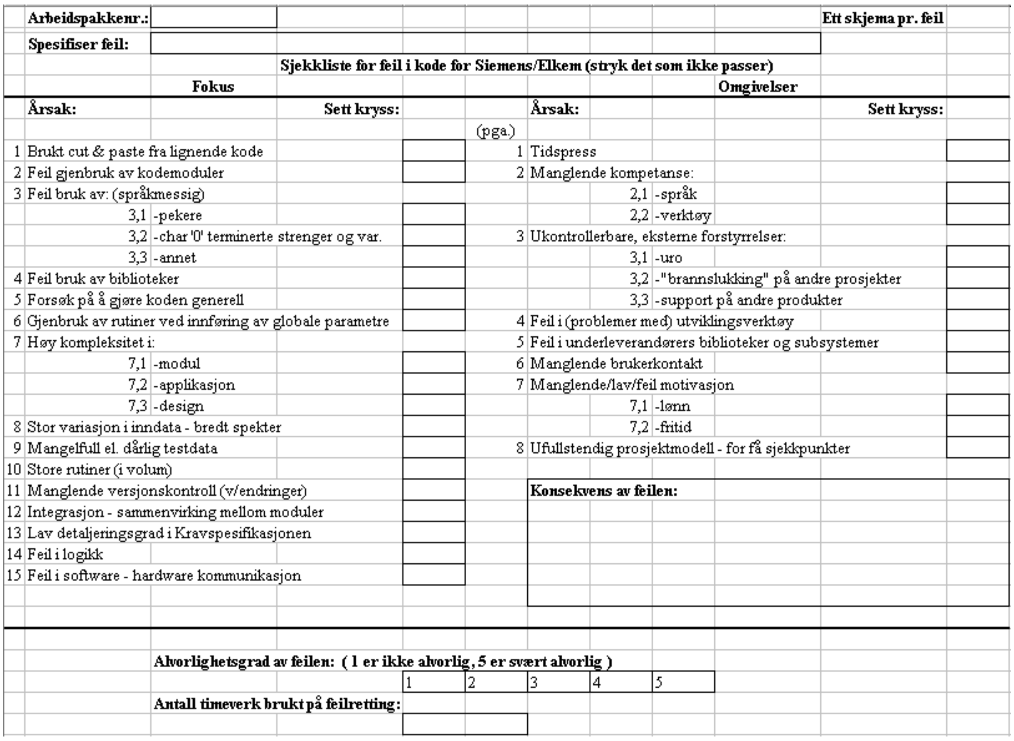
Variation factors – secondary causes:

1. Time pressure during development
2. Missing competence in use of programming language or development tools
3. Uncontrollable, external disturbances – for instance fire fighting old systems
4. Errors in development tools
5. Errors in libraries and subsystems supplied by subcontractors
6. Lack of user interaction, which caused lack of understanding of user needs
7. Lacking or missing motivation among the developers
8. Incomplete project model, for instance too few check points in the project

The item numbers used in the lists above are the same as the once used in data collection form shown in Figure 1 below.

In order to get a more detailed picture of the reported failures; some of the categories were split up into subcategories. It turned out, however, that this did not add important information, and these subcategories were dropped from the later Pareto analyses. They were, however, useful during the RCA process when filling in the Ishikawa diagram.

In addition to the possible causes, we registered for each reported failure; its degree of seriousness, consequence and resources needed to correct it. The seriousness was scored on a five point scale, the resources needed for correction was registered in person-hours, while the failure consequences were registered in free text format. The final form – in Norwegian – is shown below.



	Arbeidspakkenr.: □					**Ett skjema pr. feil**	
	Spesifiser feil: □						
	Sjekkliste for feil i kode for Siemens/Elkem (stryk det som ikke passer)						
	Fokus				**Omgivelser**		
	Årsak:		**Sett kryss:**		**Årsak:**		**Sett kryss:**
				(pga.)			
1	Brukt cut & paste fra lignende kode		□	1	Tidspress		□
2	Feil gjenbruk av kodemoduler		□	2	Manglende kompetanse:		
3	Feil bruk av: (språkmessig)				2,1	-språk	□
	3,1	-pekere	□		2,2	-verktøy	□
	3,2	-char '0' terminerte strenger og var.	□	3	Ukontrollerbare, eksterne forstyrrelser:		
	3,3	-annet	□		3,1	-uro	□
4	Feil bruk av biblioteker		□		3,2	-"brannslukking" på andre prosjekter	□
5	Forsøk på å gjøre koden generell		□		3,3	-support på andre produkter	□
6	Gjenbruk av rutiner ved innføring av globale parametre		□	4	Feil i (problemer med) utviklingsverktøy		□
7	Høy kompleksitet i:			5	Feil i underleverandørers biblioteker og subsystemer		□
	7,1	-modul	□	6	Manglende brukerkontakt		□
	7,2	-applikasjon	□	7	Manglende/lav/feil motivasjon		
	7,3	-design	□		7,1	-lønn	□
8	Stor variasjon i inndata - bredt spekter		□		7,2	-fritid	□
9	Mangelfull el. dårlig testdata		□	8	Ufullstendig prosjektmodell - for få sjekkpunkter		□
10	Store rutiner (i volum)		□				
11	Manglende versjonskontroll (v/endringer)		□		**Konsekvens av feilen:**		
12	Integrasjon - sammenvirking mellom moduler		□				
13	Lav detaljeringsgrad i Kravspesifikasjonen		□				
14	Feil i logikk		□				
15	Feil i software - hardware kommunikasjon		□				
		Alvorlighetsgrad av feilen: (1 er ikke alvorlig, 5 er svært alvorlig)					
			1	2	3	4	5
		Antall timeverk brukt på feilretting:					
			□				

Figure 1: Data collection form – in Norwegian

What is our experience

The use of GQM worked as intended and we were able to create interest and ownership for an improvement initiative. The failure cause categories that came out of the GQM process were practical and useful. A good indicator of its success is that only one failure cause was added later and none of the original failure causes were removed.

Data analyses

Pareto analyses

Vilfredo Pareto, an Italian Bachelor of Commerce who lived in the nineteenth century, did research on poverty in Italy. He discovered, from extensive statistical material, that approximately 20% of the people in Italy owned or controlled 80% of the country’s total resources. This 20/80 distribution has proved to be applicable in many different environments, among them sources of failures in software development, and is known as Pareto’s Law.

The Pareto analysis is a deceptively simple, yet powerful, way of looking at data to help find the root-cause of a problem. We used a Pareto analysis on our collection of data to identify areas of improvement rather than direct problem removal. Our goal was to identify the main problem areas so that they could serve as a basis for a root cause analysis applying Ishikawa diagrams in combination with a brainstorming session.

The Pareto diagrams are informative in the way that they visualise the main problem areas. The diagram also fits in nicely with our experience that plotting is the best way to start any data analysis process [6]. Below we show the diagrams for the two projects in the case study. This shows us which main causes are prominent in this study. The cause numbers along the x-axis in each Pareto plot refer to Figure 1.

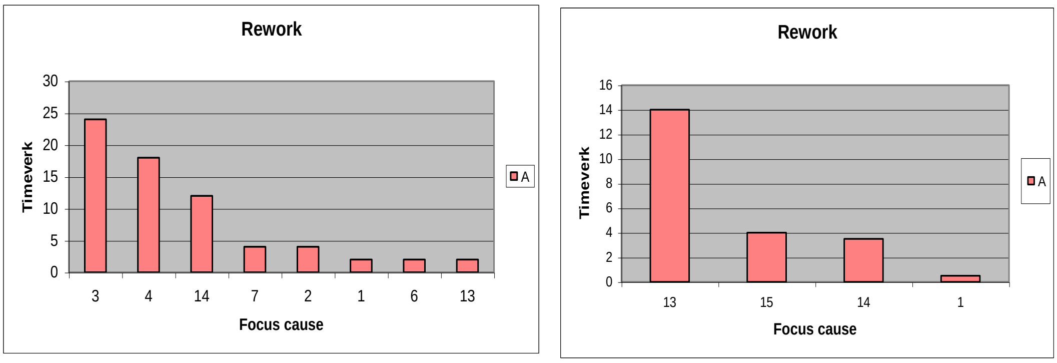


Figure 2: Pareto diagrams for the two projects. Project 1 to the left.

We found combining Pareto analyses, brainstorming sessions and Ishikawa diagrams in a root cause analysis to be an efficient method for identifying problems and analysing problem areas.

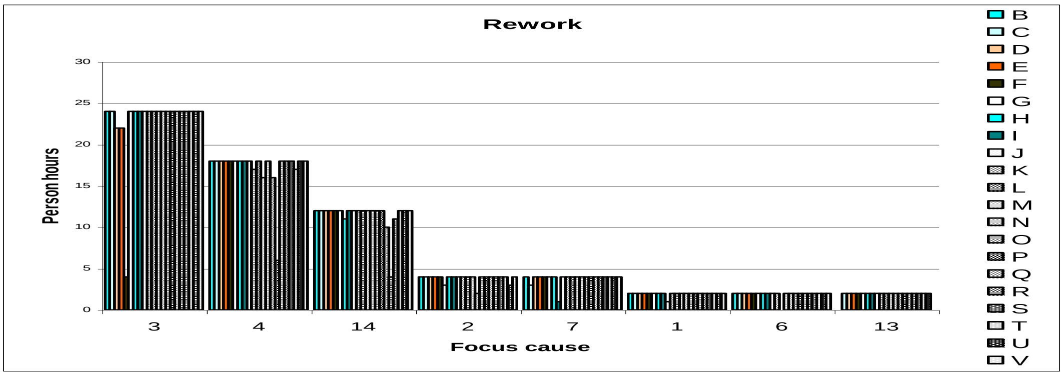
Robustness analyses

The amounts of data collected in this case study is small. The reason for this is that the company is small and with only a few projects running at the same time. We can not collect data over a long period of time, because collected data will soon be outdated or irrelevant due to changes in the company's environment. We therefore chose to extract information from the collection of data and convert it into improvement actions without collecting large amounts of data needed for a traditional statistical improvement approach.

Since we had few observations, each data point can be critical for the conclusion. We thus introduced robustness analysis to assess the confidence we should have in the data – and thus in the conclusions.

Broadly speaking, a robustness analyses is done by checking whether the conclusion changes when one of the observations was removed from the data set. To do this we created new data sets consisting of one observation less than the total data set. The amount of new data sets is equal to the total amount of reports. They are all different because there is a different observation missing in each of the new data sets. In the first new data set, we removed the first observation. In the second new data set, we put the first observation back and removed the second observation and so on. In figure 3, the letters B, C, D and so on mark these data sets.

By plotting all the new data sets into the same Pareto diagram, we can validate our conclusion. If all the columns are identical and they are identical to the original Pareto diagram based on the total data set, we have a robust conclusion. The total set of data gave us the Pareto solutions presented in figure 2. Combining all the sets from each project for the focus cause gave us the following two Pareto diagrams:



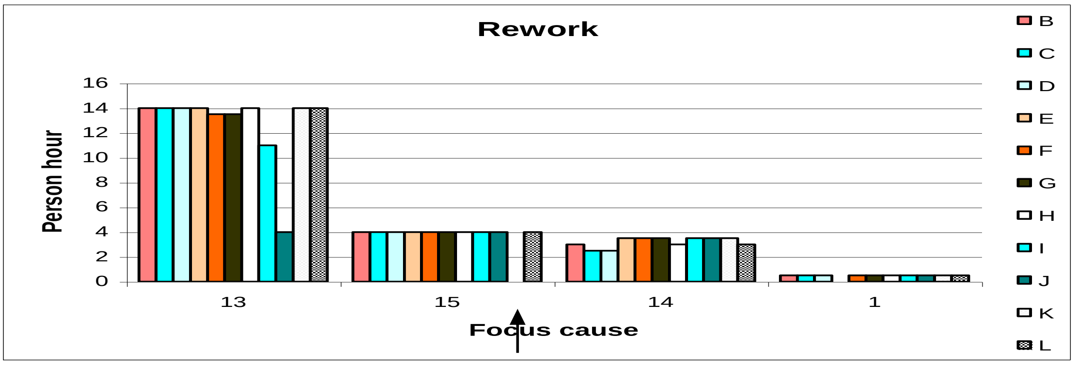


Figure 3: All data sets presented in one Pareto diagram for each project to see if it is robust. Project 1on top, Project 2 at the bottom.

In our case several of the new data sets indicated a different conclusion. The observations removed from these data sets are to be considered critical to the conclusion because of their ability to change the conclusion. An example from the graphs above are set K – indicated by an arrow - from Project 2 which show causes 13, 14 as a result instead of causes 13, 15, 14.

The number of new data sets giving us different results can tell us how much confidence we should have in our conclusions:

(Total amount of observations – amount of critical observations)/(Total amount of observations)) which gave us 81% in both projects.

The critical observations should be checked one extra time. We did this by going back to the person who filled out the error report asking him to check it. This check fortunately confirmed all the critical data.

After analysing both projects as described above, we ended up with the following main problem areas:

Project 1:

- Incorrect use of language features
- Incorrect use of library components
- Wrong code logic

Project 2:

- Incomplete or insufficiently detailed requirements specification
- Incorrect component integration
- Wrong code logic

All the identified main problem areas were then analyzed by using an Ishikawa diagram.

Root cause analysis

We had two main requirements for our choice of method for data analyses in this project. It should be simple to use with no - or almost no - training needed in advance and it should encourage and facilitate developer participation.

The result was a two pronged approach. We decided to use the Ishikawa diagram for documentation and visualisation. The use of the Ishikawa diagram should take the results from the Pareto analysis as its starting point and then be combined with general brainstorming techniques.

The Ishikawa diagram

The data analysis in the improvement project was done by using the Ishikawa diagram. There were two reasons for this: First and foremost, this method is well known in industrial activities. It is easy to understand and apply and it is possible to identify improvement possibilities once the diagram is finished. Secondly, the method is well suited for developers' participation, which we consider to be an important feature in any SPI activity.

The Ishikawa diagram has been used for many years in improvement activities in a wide variety of industrial settings. It is part of the seven old TQM tools and there is a large amount of literature that covers how the technique can be used in an efficient manner. The method is application area independent and could thus be used without any modifications or adaptations.

An example of an Ishikawa diagram is shown below. The horizontal line – the trunk - is the problem that we want to analyse. In this case it is "incomplete requirements". The main branches are filled in with the main causes as the participants see them. The next level horizontal lines represent possible causes for the main causes and so on. When the brainstorming session is completed and the Ishikawa diagram is finished, we go through all the identified causes and move or delete causes as necessary. As a last activity, the most important main causes should be identified and put up as candidates for improvement activities.

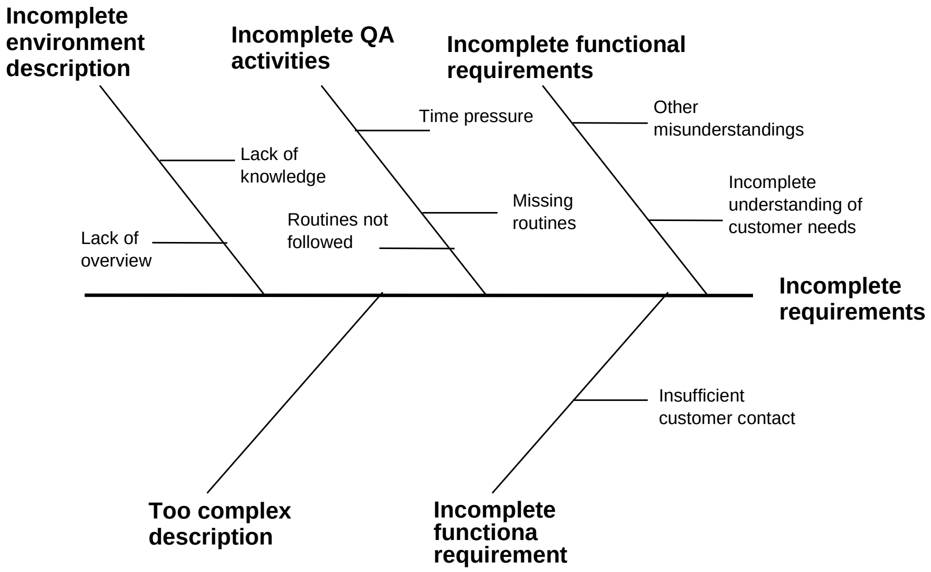


Figure 4: Cause and Effect Diagram from one of the brainstorming sessions

Brainstorming sessions

Brainstorming sessions are well organised, structured meetings integrating the project team and the measurement team. The collected data are presented to and interpreted by the project team members. Combining the knowledge of the team members with available data, we hoped to identify the root cause.

Since none of the project participants had any prior knowledge of RCA, we started with a half-hour RCA introduction. As we have observed in earlier cases, it turned out that RCA itself was easy to understand and use. During this first half-hour session we were able to describe the method and work through a simple example. One of the strong points of a brainstorming session is that it maximises the use of expert knowledge in combination with the collected data.

After this short introduction to the project team on what we expected and what we had found, we started the brainstorming session. The results from this two hours plus session were summed up in a report that was used as input to the post-mortem analyses at the end of the two development projects.

Results

The splitting up of the possible causes into focus and environment, forced upon us by GQM was of great help in setting up the Ishikawa diagram. The secondary causes identified during data collection was used as a starting point when identifying the second level horizontal lines representing possible causes for the main causes.

When we felt confident that the right root causes had been identified, we started to work on the improvement steps. This last step proved to be the easiest one. The discussion among the participants in the feedback session was already going high on what to do to improve the situation in the company.

One of the first things that came up was sharing of competence and experience within the company. Many employees with seniority have a lot of experience that can be shared with the rest if the employees. By shearing this information through courses, lectures or written reports, valuable information will be spread in the organisation. Another thing that came up was availability of information, templates and routines. The lack of an Intranet was also mentioned, and this was one of the points discussed where everybody agreed. Other improvement actions from this session are presented in the list below:

- Sharing of competence internally
- Checklist on Intranet
- Common routines/solutions for standard tasks (best practice experience)
- Common dynamic document templates
- Start the planning, design and development of an experience database

The next activity will be to look at cost and risk in implementing one or more of the identified improvement activities. This is beyond the scope of this case study but work is in progress in the company.

The success of new improvement activities will depend on the participation of all the developers in the company. Our experience is that this method of introducing process improvement to all the involved parts in an organisation increases the probability of success.

Conclusions

We started on this case study with an idea that we could do magic for the company by just waving our magic wand – the improvement framework originally proposed by the TQM fathers. It was not that simple, in fact it was not simple at all. There is a lot of “trial and error” behind this case study, but the important part is that we learned, and that the company learned.

It is important that those who shall collect the data participate in the improvement process right from the start. Introducing the participants to the necessary parts of GQM, and involving them in defining what to measure and how to measure it, motivated the participants and thereby increased our confidence in the collected data. The concrete focus on improvement we set in this session had a positive impact on the improvement culture in the organisation.

Is seems clear that we seldom or never will get enough material for solid statistical analyses in a company of this size. By introducing robustness analysis as an enhancement to the Pareto analysis, we found a way to assess the confidence we could have in the data and thereby the conclusions.

In order to identify the main causes in the development process, we held a brainstorming session supported by Ishikawa diagrams. This gave us a new opportunity to involve the project teams in a discussion on what were the main causes and what the possible causes to these main causes could be. In addition to giving us the root causes it also gave us a lot of suggestions as to what to do about it.

As a result of this case study the work started on creating an Intranet site. It was decided that a set of pages on this site should be dedicated to points on the list shown above. One person was also dedicated to keep the site alive. All together we can summon up what the company learned from this case study into:

- Individual experiences for the team members
- Measurements gave knowledge of how things really are in the company
- Common understanding of what we learned from this knowledge
- Explicit shared knowledge about the process that the company can reuse

The most important experience, however, is that it is possible to improve the process through data analyses and the use of simple problem solving techniques.

References

[1] Aune A., *Kvalitetsstyrte Bedrifter*, Ad Notam Gyldendal, Norway, 1993. ISBN 82-417-0516-6 (in Norwegian)

[2] Conradi, R., Juul Wedde, K., Stålhane, T. Sørumgård, S. m.fl., *The SPIQ Handbook* - V 1.1 05.01.1998 (in Norwegian)

[3] Lillestøl, J., *Kvalitet: Idéer og metoder – offensiv kvalitetsutvikling*. Fagbokforlaget, Norway, 1994. ISBN 82-7674-032-2 (in Norwegian)

[4] Damele, G., Bazzana, G., Andreis, F., Aquilio, F., Arnoldi, S., Pessi, E.: *Process Improvement through Root Cause Analysis*. Italtel SIT BUCT Linea UT, 1997

[5] Basili, Victor R., *Software Modelling and Measurement: The Goal/Question/Metric Paradigm.* University of Maryland Technical Report UNIACS-TR-92-96, 1992

[6] Juul Wedde, K., *Analyse og presentasjon av måledata*. SPIQ technical paper (in Norwegian)

[7] Straker, D., *A Toolbook for Quality Improvement and Problem Solving*, Prentice Hall, 1995, ISBN 0-13-746892-X

[8] Stålhane, T., *Forbedring gjennom årsaksanalyse (RCA)*. SPIQ technical paper (in Norwegian)

Error Trending

Why and How

Niels Bruun Svendsen
B-K Medical A/S, Denmark
nbs@bkmed.dk

Introduction

How do you waste your money? Do you make the perfect error free product and loose the market while doing so or do you get your product out "first thing" and drown in error corrections, patches and possibly field updates?
When developing systems and software an inevitable management question is: "When is the system ready for release?". On the bottom line the answer on when to release a new product for production and sales is a matter of being able to estimate the cost of releasing, as well as the cost of postponing the release.
In calculation of the cost of releasing a product the number of remaining unknown errors is a major factor. Therefore error detection trends during the system-testing phase have been introduced as means of estimating the number of remaining unknown errors.
This paper will share the experiences gained and the lessons learned from introducing error trending as an estimation tool and highlight the benefits found as well as the problems encountered.
The results includes not only experiences with the precision of the estimates but also, and not less interesting, the impact of error trending on the organisation. It was found that the error trend had a great value during all of the system-testing phase, and for all groups involved:

- Top-management get a more objective estimate on remaining unknown errors
- Project managers gets a management and planning tool and arguments (against sales and top-management) for not being able to release "tomorrow"
- System test staff gets a planning tool and the arguments for extra resources
- The developers get the argument for not being able to start on another project "tomorrow", i.e.: “When this many errors are suspected to be found, we need time to fix them!”

In short this means that one simple curve gives input and insight for top-management, project management, QA function and developers, i.e. becomes the common reference on system state.

Company Context

B-K Medical develops, produces and markets ultrasound systems for medical diagnostic imaging. The systems are sold throughout the world with the major markets being Europe, USA and Asia. B-K Medical has 250 employees with 166 located in Denmark. The development department consists of 60 employees where 20 are involved in software development. B-K Medical is ISO 9001 certified and most of the products have FDA market clearance and are CE-Medical Device certified. Therefore external audits are performed accordingly. No formal assessment against a model has been performed, but an informal self-assessment using the BootCheck tool from ESI has been performed. This assessment gave maturity ratings between 2.5 and 3.25, indicating some areas in need of improvement to get to the Defined (3) level, and a general lack of metrics as required in the Managed (4) level.

Project Initiation

The introduction of error trending at B-K was initiated by a management request for an improved basis for making the release decision, i.e. to decide whether or not to release a new product for production and sales. As part of the initiatives taken in order to pursue this goal, Error Trending was introduced. By using Error Trending to estimate the number of remaining unknown errors rather than using pure intuition, the objectivity of the basis for the release decision is increased.
Although aiming primarily on an estimate of remaining errors at the time of the release decision, error trending was introduced in the system-testing group as a tool to be used from the beginning of system test execution until the product is released. Beginning error trending early in the system-testing phase gave a lot of good experiences as described later on.

The initial steps with error trending were done on error data from a scanner that had been on the market for a year and therefore the number of reported error after release was known. Error reports from the last part of the system-testing phase were used and plotted as seen in fig.1. The y-axis shows the accumulated number of errors reported, and the x-axis shows the number of test days. A test day is equal to a calendar day except that only calendar days where test were performed are included.

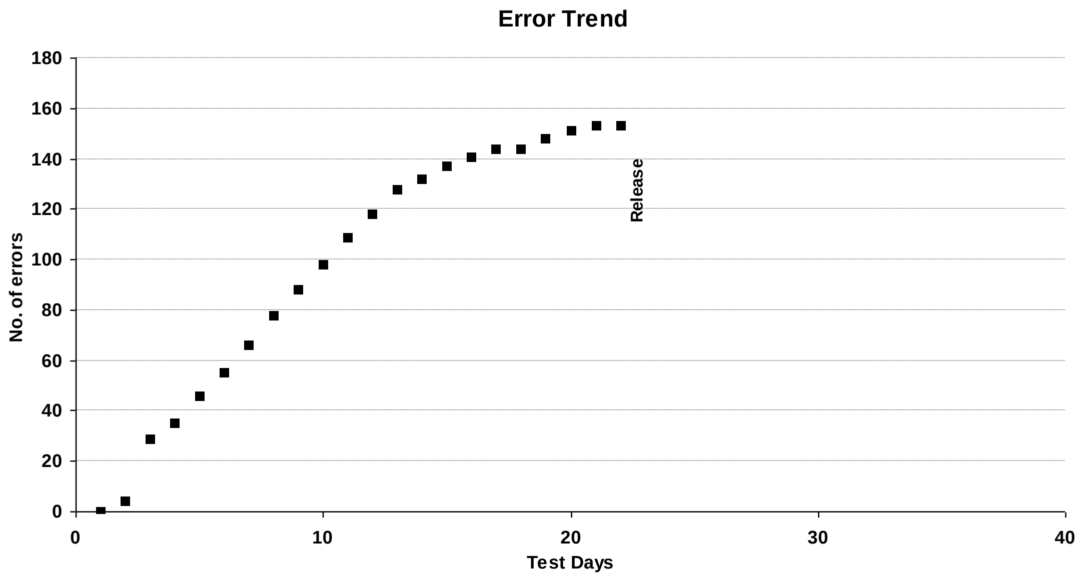


Fig.NBS.1 : Accumulated no. of errors for released product

Despite the fact that the test effort pr. test day was not known in any detail, the plotted error data gave a quite clear trend with a distinct convergence in the last part of the trend. To start off with, very simple functions were tried out, using the trending functions in the MS Excel spreadsheet. None of the experiments using all data gave any trustworthy results. Our criteria for a result to be trustworthy, were that the estimated trend had a good correlation with the last converging part of the data, and that it gave an estimated total number of errors higher than the number of errors already found.

Finally it was decided to focus only on the latter part of the error data, and using the exponential function on those data as shown on fig. 2. It gave a perfect match with the number of errors actually found after release.

Although this was very well affected by the fact that we knew the result we should get, it did give some confidence in that here was something useful. Fig. 2 was used for raising internal interest in error trending, with the argument that:

> Based on data with a great deal of uncertainty you can apparently still draw and extrapolate a trend using the data from the final stage of system test and get a very good estimate on remaining errors.

The conclusions on the work with the data from the released product was that although limited in amount and precision it gave a good initial interest in error trending and was a kick-off for going further into the subject.

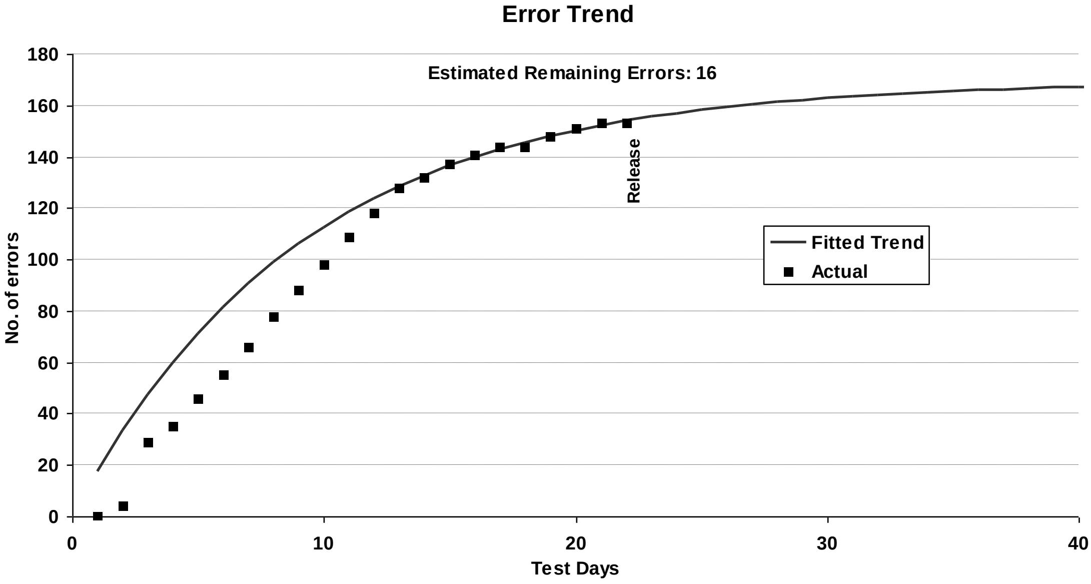


Fig.NBS.2 : Exponential Error Trend for released product

The Model

When searching for experiences on error trending the name of SATC (Software Assurance Technology Center) at NASA is very likely to pop up. SATC has published articles that mentioning their work on an Error Trending Model, ref.[1] & ref.[2]. The Error Trending model was also mentioned by Linda Rosenberg, SATC at a QWE'98 tutorial. As we did not find any further description of this model, Linda Rosenberg was contacted. We got a very quick response saying that work was still in progress on the model and they were working on a tool to support the model. We were also invited to send our data to SATC to have them analysed.
We decided to send data from the first part of system testing on a new scanner. Unfortunately our data did not give any valid results when analysed by the SATC Software Error Trending Tool. Instead they returned a spreadsheet with our data analysed by a Weibull variate. It differed from the Weibull function in relation to the manpower utilisation, but as this did not influence the estimate on the number of remaining errors, we decided to proceed with the Weibull function itself. The Weibull function has the form:

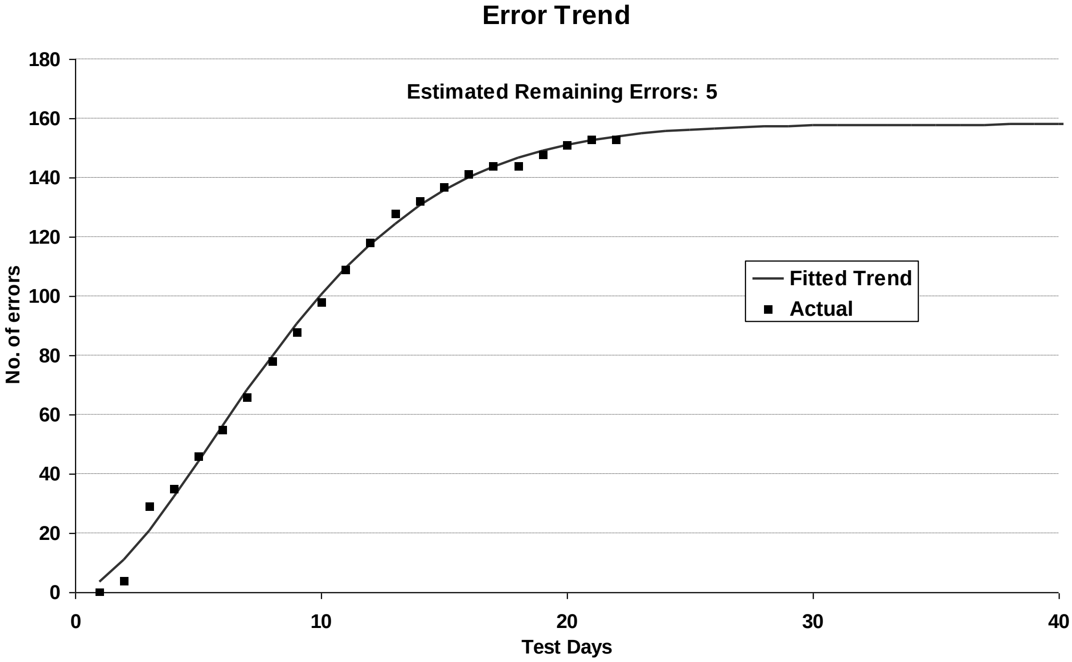
$$f(t) = K * \left(1 - \exp\left[-\left(\frac{t}{t_{\max}}\right)^p\right]\right)$$

With p = 1, we have the exponential function and with p = 2, we have the Rayleigh curve. When used for trending, the parameters k, tmax and p are optimised to get the minimum sum-of-difference-squared. The spreadsheet included set-up for using the MS Excel solver to analyse additional data, and has formed the basis of our further work with error trending. We are therefore very thankful for this valuable input from SATC.

The Weibull function and the alternative models that could be used are described further in reference [3], [4] and [5].

In fig. 3 the use of the Weibull function on the data from the released scanner is shown. The estimated number of errors remaining is 5. A total of 15 error reports have been made since release, including also change requests.

Fig.NBS.3 : Weibull Error Trend for released product



Multiple reasons for the difference can be and has been discussed, e.g. was the system test as thorough as the "test" performed by having customer using the system, did we have sufficient data to make a reliable estimate etc. In this case the found errors were not corrected, otherwise errors introduced while making corrections could have been a reason.

The conclusion drawn on the estimate was that although a bit low, it is still a good estimate, especially when taking into account the uncertainty and limited amount of data. The estimate indicates a system ready for release and the errors found after release was also within acceptable level.

Error Trending during System Test

As mentioned earlier, data from the first part of system test on a new scanner were

analysed by SATC, NASA. The results, based on the Weibull function, gave a very high estimate on the number of remaining errors, as well as a high number of days to find the remaining errors.
When presented for the project manager we had the first direct impact on the project:

With that many test days left, we need more test objects

The presented error trend and estimates were the direct cause for additional test objects to be arranged for. The fact that the calculations on our data were made by NASA was used to increase confidence in the estimate.

A good reference gives confidence

From this point in the system test phase, daily updates of the trend and estimates were made, i.e. yesterdays reported errors were entered and new parameters for the Weibull function were calculated. The test days are here counted as test man-days, e.g. 3 testers working one day, results in 3 test days. This way we account for the changes in test effort.

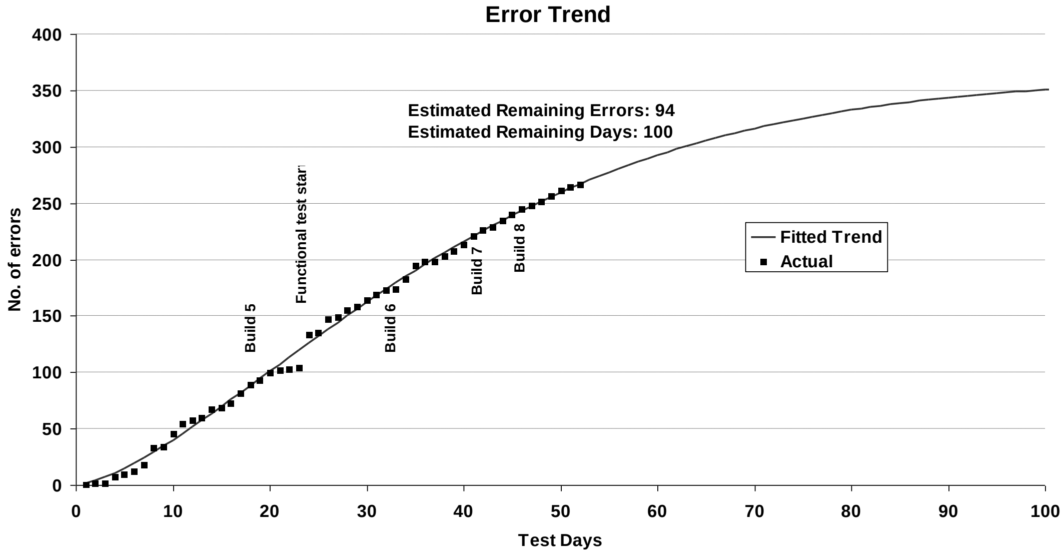


Fig.NBS.4 : Weibull Error Trend for new product

The new trend and estimates were presented on the "project wall", and on the Intranet, see fig. 4. A lot of internal interest were gained and although not all understood that the error trend curve were optimised every day, it gave opportunities to discuss the state of system under test as well as error trending in general.

During the last part of the system test phase the project manager had a demonstration of the system for the top-management. A full functioning scanner was demonstrated and as often in these situations the comment that the project manager receives is: "This scanner looks complete. Why don't we release tomorrow or at least at the end of the week?". The standard answer to this question is that "we still need a little optimisation on the quality of the image" and "we haven't got all parts in production quantities".

But this time the project manager had another argument, i.e. the error trend and the estimate of remaining errors and test days. So he showed the error trend saying: "See we estimate the need for another 100 test days. With the number of scanners and testers we have, that means we're finished in 30 days, and that is exactly the planned release date. That was very convincing and made the end of that discussion. Of cause the input from SATC at NASA again played a role in the creation of confidence in the estimate.

The product is finished. Why not release "tomorrow"?
The Error Trend holds the answer

This time it was the top-management, but next time it will be the sales staff asking for a release "tomorrow". The visualisation of the Error Trend makes it easy to communicate the probability for further errors to all types of staff in the company.

Not only the project manager, but also the developers can make use of the Error Trend in this stage of the project. Typically another project is crying out for development resources as soon as they have finished their work on the current project. And there is a strong tendency for developers to be almost finished, i.e. "I have only a few more (known) errors to correct, then I'm finished. A few errors might pop up but we'll fix them in-between the other work". Here the Error Trend is a great help too as it is easy to take the number of estimated remaining errors and divide by the number of developers and you have an estimate on how many more errors there are to correct for each developer. In our case and probably for many others, this will mean a considerable amount of time to be planned for before the resources are ready for the next project.

You have implemented it all. Why can't you start on a new project "tomorrow"?
The Error Trend holds the answer

The value of the Error Trend and the estimates in the mentioned situations naturally depends on the precision of the estimates. However we find that the normal expectations are that far from any reality, that almost any estimate is better than none. The benefit is there if just you can show that there is "a lot" of errors left and not just "a few".

Fluctuations of the Error Trend

Fluctuations in the trend was expected, as new builds, new test techniques and the start of test in previously untested areas are very likely to initially increase the number of errors found. And when you test on the same build with the same test technique fewer and fewer errors will be found. This phenomenon is seen in fig. 4, where the first 24 test days constitutes its own "S" curve and a large increase in error detection rate is seen as we enter what is referred to as functional test.

So fluctuations are seen:

- When test of new features is started

- When changing test techniques
- When new builds are introduced

In our case the largest fluctuations were seen when entering test of new feature and the smallest fluctuation seen when introducing new builds.

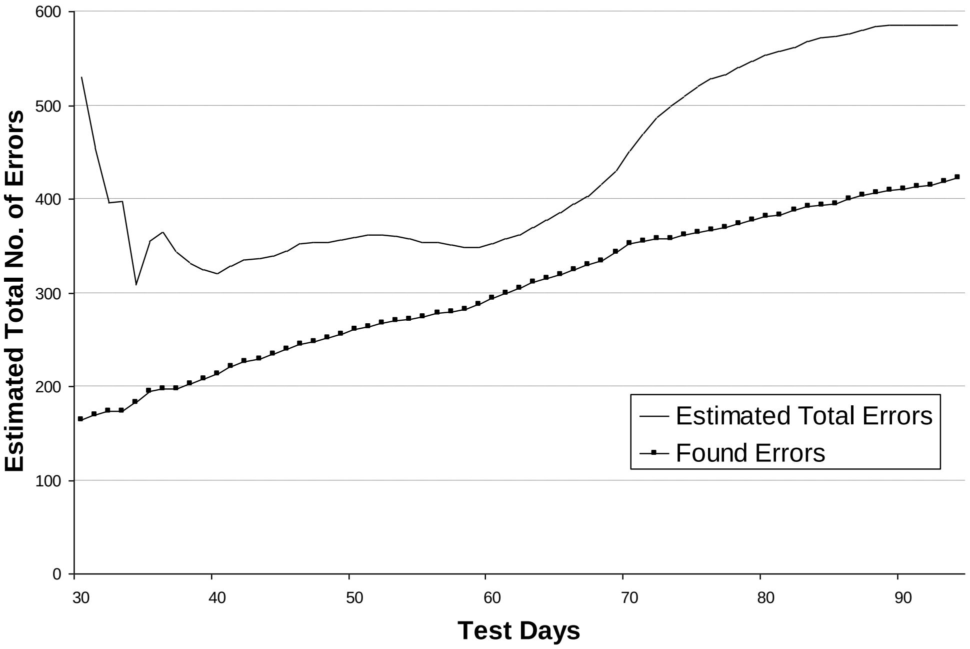
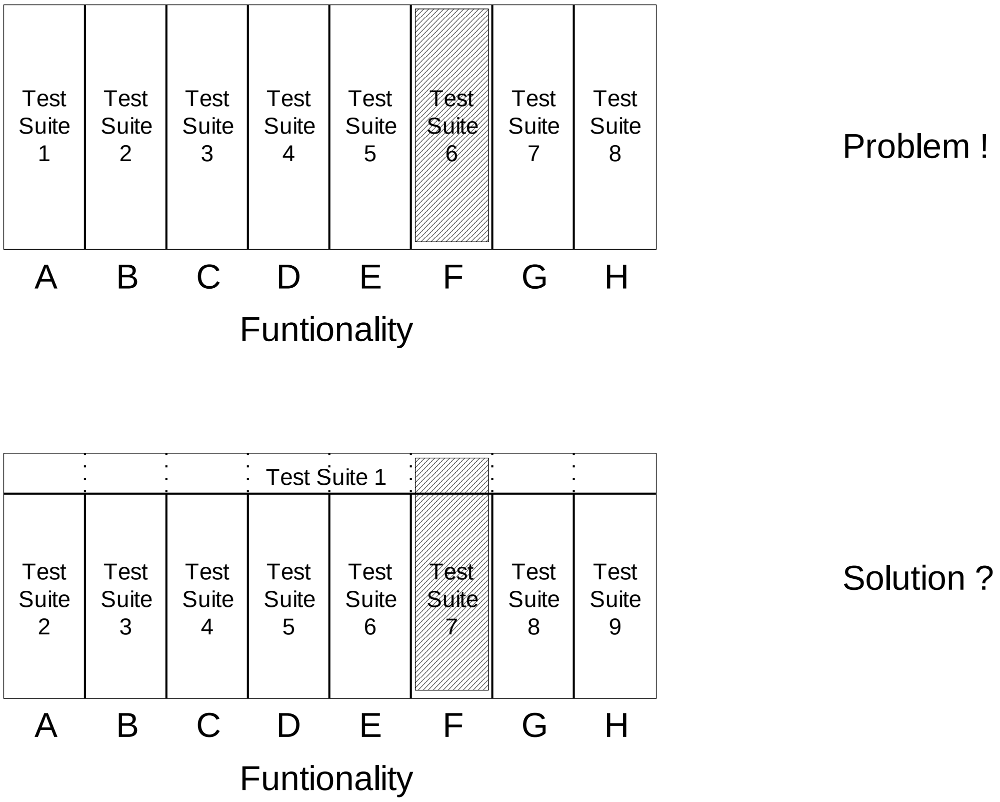


Fig.NBS.5 : Evolvement of estimated total no. of errors

As the estimated total number of errors were calculated every day these fluctuations had an impact on the estimated total number of errors. Therefore there was a need for visualisation of the evolvement of this estimate. A trend for the estimated total number of errors was added as seen in fig. 5. The first estimate of 530 errors in total was the estimate received from SATC's analysis of our data and the figure used to get additional test objects. As seen the estimate was reduced somewhat during the first period where Error Trending was used and we saw the estimate stabilise around approx. 350 errors. But then around the 65th test day suddenly the estimates of the total number of errors increased drastically. This was caused by the fact that we had entered test of 2 previously untested areas that were found to have a much higher error density than what had been tested so far.

This increase in the estimated number of errors in the system naturally imposed a problem on the project, both in getting development resources to correct the errors and the extra time needed for both the correction and the verification of the corrections. When discussing the situation we could see that this was not a new problem, but a problem we have had "always". It is a result of the way we plan the system test, where we execute the test sequentially, function by function. The problem is visualised in fig.6. The illustration shows a set of functionalities, where the “F” functionality is significantly more error prone than the others. The first case is how we traditionally have covered the test of such a system with test suites for each functionality and executing the test suites sequentially. This means that we will not have any knowledge of the, in this case, high error density of “F” until late in the system test execution phase.

Fig.NBS.6 : Test Sequences



Therefore we have changed the strategy for test planning slightly, making a test suite that covers all functionalities. This test suite will not cover any functionality in depth, but just enough to get an impression of the error density of the functionality. Use Cases will be used for designing this test suite. By executing the Use Case based test suite as the first test suite, we will get valuable data for planning the execution of the remaining test suites. We will also get the possibility to reject functionalities early in the test process, limiting the time spend on system testing features that are not ready for system test. This way of planning the execution of the system test will be applied in two projects during autumn 1999.

Common Sense has to be triggered

This change in the system test execution is not directly connected to the Error Trending. But the visualisation of the problem that the Error Trend caused was the trigger needed to realise it and to have a broader group of people discussing the problem and possible solutions.

In the final stage of the system test it was found that the estimated total number of errors remained at a very high level, even with many test days having no or very few errors found. When looking at the trend curve it was apparent that it was not following the actual error findings in the final stage of the system test very well. Therefore the initial part of the system test, where new features were still added to the system, was omitted from the trend calculations in the final stage of the system test. So just as there were valuable impact of starting Error Trending early in the system test, even though not all of the system was ready, it was found that the estimates to be used in the latter part of system test had to be based solely on data starting at the time where the total system is available.

Summary

The experiences with the work performed with Error Trending can be summarised in the following Why's and How's:

Why:

- It triggers the use of common sense
  - It highlights the need for process improvements
- It's a valuable tool implemented by simple means
- It works as a common reference on system state
  - Managers gets valuable knowledge
  - Developers gets valuable knowledge
  - Testers gets valuable knowledge
- It improves the release decision support

How:

- Find and plot available data
- Select an error trend approach
  - Ideas can e.g. be found in literature by SATC, J.D.Musa, Grove Consultants and S.H.Kan
- Apply it during system test
  - Make it visible to all involved in the project
  - Monitor the trend and learn from the questions and discussions it generates
- Do not initially expect high-precision estimates
  - Increase accuracy by process improvement actions

Conclusions

We started off aiming at a technique to estimate the number of remaining errors at the time of possible release. What we found was a technique that apart from doing that were able to trigger common sense in several processes related to the system test phase. Improvement was triggered in relation to:

- Being realistic about when development resources are ready for the next project
- Getting an easily communicable argument for not releasing "tomorrow"
- Planning the system test for early error density overview

So far we have limited data on the precision we can obtain, but we have found that even with a limited precision there's a lot of benefit in collecting and presenting data which in many cases are fairly easy to get hold of.
Apart from the mentioned models we also tried using 3rd order polynomial approximation as suggested by Grove, but had some problems getting estimates we believe in. And the trust in the model is a key issue when the idea is to be "sold" internally. Also a good reference play a key role in that respect. But whatever model you choose, don't trust it blindly. Keep your common sense and professional knowledge, but let Error Trending help you stay objective and use it as a mean of communicating between personnel groups.

Get your Error Trending started – You won't regret it

References

[1] Robert E. Waterman, Lawrence E. Hyatt : "Testing - When Do I Stop?" International Testing and Evaluation Conference, Washington, DC - October, 1994

[2] Dr. Linda Rosenberg, Ted Hammer, Jack Shaw: "Software Metrics and Reliability"
9th International Symposium on Software Reliability Engineering Germany - Nov 1998

[3] Stephen H. Kan: "Metrics and Models in Software Quality Engineering", Addison Wesley, ISBN 0-201-63339-6

[4] J.D.Musa, A.Iannino, K.Okumoto: "Software Reliability: Measurement, Prediction, Application, ISBN 0-07-044093-X

[5] J.D.Musa: "Software Reliability Engineering, More Reliable Software, Faster Development and Testing, ISBN 0-07-913271-5

Session 12
SPI and People

Chairman
Tor Stalhane

Sintef, Trondheim, Norway

TEI R&D-GPC People Management and Development Process to motivate, develop and retain the best resources

G.Evangelisti
Ericsson
Telecomunicazioni SpA,
Roma, Italy

E. Peciola,
Ericsson
Telecomunicazioni
SpA,
Roma, Italy

C.Zotti
Ericsson
Telecomunicazioni
SpA,
Roma, Italy

Abstract

"The only rule I have in management is to ensure that I have good people-real good people-and that I grow good people, and that I provide an environment where good people can produce"

Software vice president quoted in (Curtis88)

In this paper we report about the results of a program run in Ericsson Telecomunicazioni Italia, Research & Development – Global Product Center Division (**TEI R&D-GPC)** aiming to the definition of a process for People Management.

The process describes an innovative way to handle human resources in order to:

have motivated people

realise an effective job staffing

prepare the organisation for the future.

The process was launched in May 98 and we report preliminary feedback on its application based on data collected from a questionnaire measuring people satisfaction distributed to all the staff in December 98.

Introduction

In the last 20 years Software organisations have been concentrated in the improvement of their capability working actively on processes and technologies.

But this is not enough to run ahead of the continuous change: the software systems are more complex, the competition between companies is stronger and the customer demands develop much faster, aiming at lower costs and efforts and at an higher quality.

In the 90's many organisations discovered that, in the frame of software improvement, processes and technologies are not enough and that also a third component, the people, plays an important role, as depicted in Fig. 1.

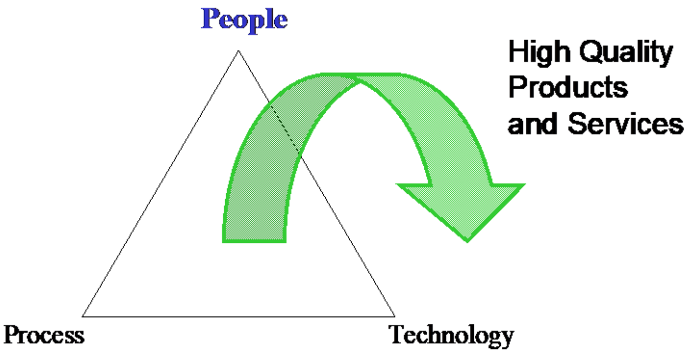


Fig. 1: Three Component of Improvement Focus

The demand is to focus on methods for improve the way people-related issues are addressed, in order to attract, develop, motivate, organise, and retain the talents needed by the organisation.

This paper wants to stress the importance of paying attention to people management, not by giving more money to some individuals, but rather by spending more in the organisational issues that the personnel show to be sensitive to.

The personnel wants to build a specific development path, to have access to interfaces to ask questions, to be supported in changing job, to have opportunities to speak with the managers, to receive more information, to improve the quality of their life during working hours (while the free time hours decrease and will decrease even more).

And they expect that attention is not paid only to managerial roles, but to each role, as each individual, independently from his/her role wants to feel that the working place can offer him/her the best development conditions.

To answer these demands an action program was started in TEI R&D-GPC during 1998, whose goals were to define a process for handling Human Resources in an innovative way, where competence is not just a matter for the individuals, but an issue for the organisation.

Goals of the Action Program

Before running this action program, People Management practices were not sufficiently formalised and the only processes regularly executed were Competence Development and Potential Evaluation.

Managers were not supported enough by the organisation in their inter-work with their people.

Carrier development of technical persons was not cared enough and their changes of job were often tied only to events rather than also to plans.

In this context, the level of services provided by the organisation was perceived by the personnel as quite low.

If we refer to a scale of layers in caring people management services:

physical environment

regulatory environment

processes and services

relations environment

we scarcely reached the second level as even the knowledge of many existing rules was not spread enough.

The goals of the action program were mainly two:

to establish practices able to give more care to the main expectations of individuals in their inter-work with the organisation

to focus on technical roles and to ensure them equal dignity as the managerial ones

In the beginning of 1998, the design of the “People Management and Development Process” (**PMDP**) was the way locally chosen to attract, develop, motivate, organise and retain the talents that were needed in TEI R&D-GPC.

To achieve the second goal, in the last part of 1998, a new method for the classification of people competence through roles was introduced.

Roles were classified into 3 ladders,

- Professional
- Solution manager

- Organisation manager

each aiming at a different organisational goal: competence, improvement, asset management.

Equal development possibilities and recognition were assured to roles independently from the ladder where they appear. The structure in levels of the three ladders allows to compare roles with different professional content and aiming at different organisational goals.

The classification in ladders does not prevent transitions between ladders, but rather facilitates the identification of competencies and attitudes that are crucial in changing role, along the individual career path.

This action program was started on the basis of the results of the questionnaire measuring people satisfaction distributed in beginning of 1996. The main reasons of people dissatisfactions were related to:

- lacks in the planning of development programs
- low level of feedback on the performed activities
- training programs partially defined
- no possibility for job rotation
- unclear criteria for development opportunities and rewarding

Competence model and roles classification

The competence model used in TEI R&D-GPC for competence management activity is shown in Fig. 2.

The activities of the company are modelled in "company processes" (e.g. Product Management Process, Provisioning Process, Marketing Process, Sales Process, Accounting Process, Human resources Process, etc.)

The "company processes " are composed of "job families", and "job families" of "roles".

At any time, each person of TEI R&D-GPC has a single role associated to him/her.

The roles are characterised by:

- a list of tasks and responsibilities
- a "reference competence radar" (competence chart).
- a "competence group"

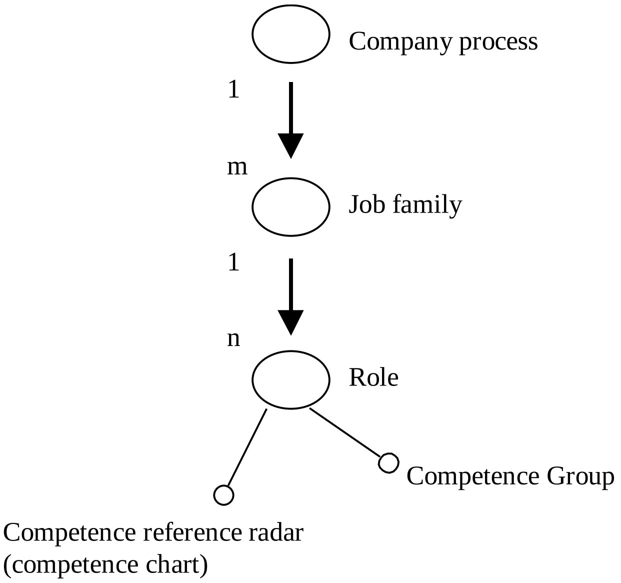


Fig. 2: The competence Model

Roles are classified into 3 ladders:

PROFESSIONAL
The responsibility consists in the provision of professional performance.

SOLUTION MANAGER
The responsibility consists in taking decisions, in negotiating, in exploiting resources, pursuing short/mid term objectives.

ORGANIZATION MANAGER
The responsibility mainly consists in managing any type of resources as company assets, pursuing also strategic objectives

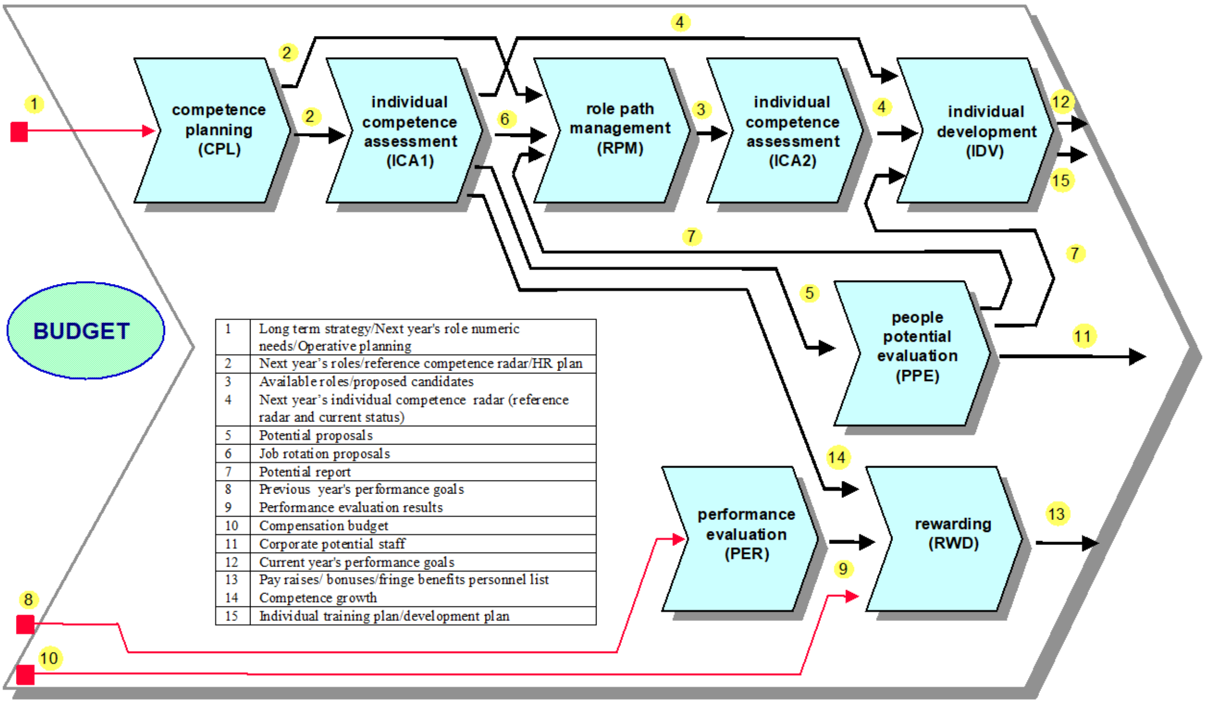
- Classification of the roles within the ladders is shown in the table below:
- To note that it is possible to reach the highest level of recognition (level 5) from each of the three levels.

Level 5 **DIR**	Expert Senior (except System) Expert (except System)	Communication Manager Master Change Manager Master Innovation Leader Master System Expert Senior Communication Manager Senior Change Manager Senior Innovation Leader Senior System Expert Product Manager Principal Product MKT Manager Principal Sys Manager Master Project Manager Principal Operation & Business Control Mgr Product Prov. Area Manager Master Mentor Master	Deputy Site Manager Deputy Assistant Manager Product Unit (PU)/Sub PU Manager Senior Design House Manager Line Manager Senior Product Mgt Line Manager Senior Product Unit(PU)/Sub PU Manager
Level 4 **QS**	Controller Master MKT Specialist Principal Specialist Master (except System)	Communication Manager Change Manager Innovation Leader System Specialist Master Personnel Officer Master Product Manager Master Product MKT Manager Master Sys Manager Senior Tech. Coordinator Master Mentor Senior Project Manager Master Product Prov. Area Manager Senior	Line Manager Product Mgt Line Manager
Level 3 **Q**	Controller Senior MKT Specialist Master Specialist Senior (except System)	System Specialist Senior Personnel Officer Senior Product Manager Senior Product MKT Manager Senior Sys Manager Tech. Coordinator Senior Mentor Proj. Manager Senior Product Prov. Area Manager	Design Team Leader Master
Level 3 **7**	Controller MKT Specialist Senior Specialist(except System) Adm. Specialist Master	Personnel Officer Product Manager Product MKT Manager System Specialist Tech. Coordinator Proj. Manager	Design Team Leader Senior Adm.Team Leader Senior
Level 1 **<=6**	Controller Assistant Team engineer senior MKT Specialist Adm. Specialist Senior Team engineer Adm. Specialist Adm.Support/Secretary		Design Team Leader Adm.Team Leader
	PROFESSIONAL	SOLUTION MGR.	ORGANIZATION MGR.
	LADDER 1	**LADDER 2**	**LADDER 3**

PMDP Description

The process is organised in eight sub-processes (Fig. 3) :

Competence Planning (CPL)
Individual Competence Assessment, first run (ICA1);
Role Path Management (RPM);
Individual Competence Assessment, second run (ICA2);
Individual Development (IDV);
People Potential Evaluation (PPE);
Rewarding (RWD);
Performance Evaluation (PER).


1
competence planning (CPL)
2
2
individual competence assessment (ICA1)
6
role path management (RPM)
3
4
individual competence assessment (ICA2)
4
individual development (IDV)
12
15
7
7
BUDGET
5
people potential evaluation (PPE)
11
14
performance evaluation (PER)
9
rewarding (RWD)
13
8
10

1	Long term strategy/Next year's role numeric needs/Operative planning
2	Next year's roles/reference competence radar/HR plan
3 4	Available roles/proposed candidates Next year's individual competence radar (reference radar and current status)
5	Potential proposals
6	Job rotation proposals
7	Potential report
8 9	Previous year's performance goals Performance evaluation results
10	Compensation budget
11	Corporate potential staff
12	Current year's performance goals
13 14	Pay raises/ bonuses/fringe benefits personnel list Competence growth
15	Individual training plan/development plan

Fig. 3: PMDP Workflow

The key issues of each sub-process are listed below:

Competence Planning

The Competence Planning sub-process aims at :

having a clear picture of the "company processes", "job families", "competence groups", and "roles" requested by the Division for the next year (definition of a human resources plan);

defining the reference competence radar for each role;

- providing an indication of the competence gap (role numerical gap) existing within TEI R&D-GPC between current year's end role staffing and next year's role numerical needs.
- The management team of TEI R&D-GPC needs this picture in order to perform the activities requested to close the competence gap.

Individual Competence Assessment (phase 1)

The Individual Competence Assessment sub-process is devoted to the assessment of the competence level of the people working at TEI R&D-GPC in order to create a reference frame for the effective management of these competencies. The Individual Competence Assessment aims at:

having a clear picture of the competencies of each person within TEI R&D-GPC;

identifying candidates for open positions so as to improve people motivation and effective staffing;

identifying candidates for people potential evaluation in order to manage them appropriately;

evaluating the competence growth of each person with the purpose of recognising his/her professional effort during the last year.

Role Path Management

The competence growth of TEI R&D-GPC personnel is supported by job rotation activities defined in this process. The Role Path Management sub-process aims at:

- defining the possible paths from each role within T division.
- providing job rotation proposals that are based both on the needs coming from Competence Planning and the candidates outcoming from Individual Competence Assessment (first run);

Individual Competence Assessment (phase 2)

In order to secure TEI R&D-GPC with critical and strategic competencies, each role in the Organisation needs to be covered by personnel with an adequate competence profile. The Individual Competence Assessment (phase 2) sub-process is devoted to the evaluation and selection of the candidates for job rotation identified in the RPM sub-process. ICA2 aims at:

- assessing the competence level of candidates for job rotation in order to select the most suitable people for the target jobs to be staffed;
- Identifying the people willing to move to other sites/divisions for overstaffed roles.

Individual Development

In order to secure that all people have the competence required to perform their assignment, each level of the Organisation needs to identify the competence required to perform critical tasks and the training needs within each line. The purpose is to ensure that needed training is received by each person. The Individual Development sub-process aims at:

- identifying an adequate competence development plan for each person;
- identifying a consistent training program for each person;
- defining performance goals for T Division first and second level managers.

People Potential Evaluation

The Organisation intends to identify and value those people that are suitable to assume key roles in the future so as to improve its. The first step, in order to achieve this goal, is the identification of people whose potential has some unexpressed dimensions. The People Potential Evaluation sub-process aims at:

- evaluating the technical and managerial potential of the candidates proposed in the list coming from the Individual Competence Assessment (first run);
- identifying which candidates can be considered potentially suitable to assume, in the short and medium term, key roles within TEI R&D-GPC, TEI and Ericsson Corporate.

Performance Evaluation

In order to grant TEI R&D-GPC first and second level managers with rewards based on their contribution to results and values, it is necessary to provide objective performance evaluation results. The purpose of PER is to produce these objective results as input to RWD. The Performance Evaluation sub-process aims at:

- assessing the performances of first and second level managers;
- summarising the evaluation of first and second level managers performances in a performance evaluation result.

Rewarding

The purpose of the Rewarding sub-process is to provide all people with rewards based on their contribution to results and values for the organisation. The Rewarding sub-process aims at

- compensating people with pay raises and level promotions;
- compensating teams with bonuses.

Description of process-time relationship

This paragraph is devoted to the relationship between PMDP sub-processes and the time. The relationship is illustrated by means of Fig. 4, “PMDP sub-processes - time relationship”. The figure considers a time window of three years: the current year - in which the processes are activated - the previous year and the next year. Time intervals pertinent to different process activities are represented by arrows.

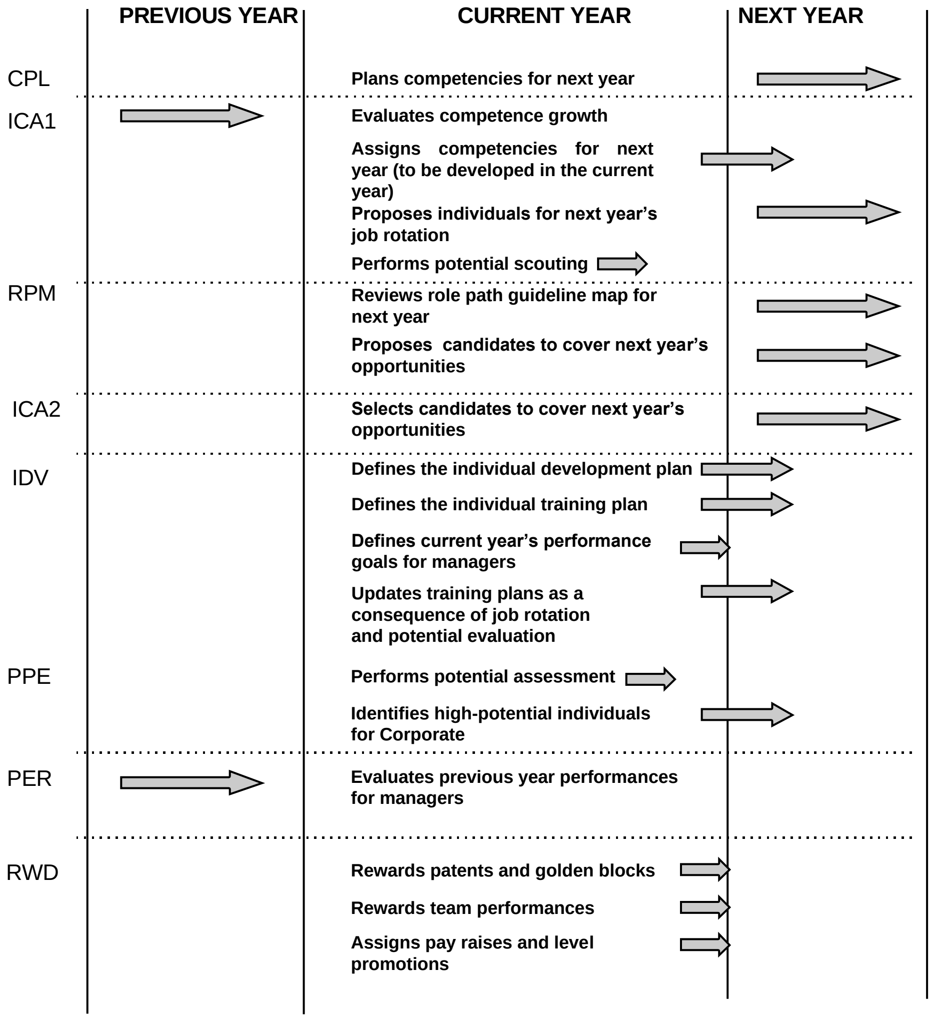

PREVIOUS YEAR
CURRENT YEAR
NEXT YEAR
CPL
Plans competencies for next year
ICA1
Evaluates competence growth
Assigns competencies for next year (to be developed in the current year)
Proposes individuals for next year's job rotation
Performs potential scouting
RPM
Reviews role path guideline map for next year
Proposes candidates to cover next year's opportunities
ICA2
Selects candidates to cover next year's opportunities
IDV
Defines the individual development plan
Defines the individual training plan
Defines current year's performance goals for managers
Updates training plans as a consequence of job rotation and potential evaluation
PPE
Performs potential assessment
Identifies high-potential individuals for Corporate
PER
Evaluates previous year performances for managers
RWD
Rewards patents and golden blocks
Rewards team performances
Assigns pay raises and level promotions

Fig. 4: PMDP sub-processes – time relationship

At the *end of January*, the **CPL** process is able to produce the first rough estimation of the next year's competence needs (i.e. the role numerical needs for the next year). This information is composed of two different data: new possible roles that should be introduced in TEI R&D-GPC and numeric gap information for each role. The first estimation of the competence needs allows the activation of the "core step" for the achievement of overall PMDP goals: the individual interview.

In *February* and in *March*, Line Managers perform individual interviews with their staff. In the same period, second level managers are interviewed by first level managers and these are interviewed by the Director of TEI R&D-GPC. The individual interviews activate different PMDP sub-processes: **ICA1**, **IDV** and **PER**. First of all, the **ICA1** deals with the competence assessment and competence growth evaluation of each person. In addition, the **ICA1** performs potential scouting (i.e. identifies people suitable for potential evaluation) and identifies possible proposals for job rotation.

The individual interview activates also the **IDV** in which the development plan and the training plan for each person are produced. The **IDV** identifies individual performance goals for first and second level managers. It is worthwhile noticing that the **IDV** also includes some work Line Manager do after the completion of the individual interviews (e.g. final training definition) and therefore this sub-process is supposed to end later than ICA1 which terminates with the end of the interviews. The scope of the **PER** is limited to first and second level managers. It is activated during the individual interview and has the objective of evaluating managers' performances by assessing the achievement of the individual goals defined in the previous year's **IDV**.

In *April*, **PPE** starts. It aims at evaluating the potential of the candidates proposed by the **ICA1**. The potential assessment allows the identification of people that are suitable to assume key roles within the Organisation, also at Corporate level. The output of the **PPE** is formalised in a "potential report".

In *July*, **CPL** is able to produce a new and elaborate version of the Human Resource Plan. This document contains information which is relevant for the **RPM**.

From *June* to *August* **RPM** performs two kind of activity. The first one is the periodic review of the guideline that defines the possible role paths in the T Division. The second one is related to the job rotation. In case the gap for a specific role is positive, the **RPM** identifies the most suitable people, among those individuated by **ICA1**, for that specific role and forwards to **ICA2** a list of job rotation proposals, i.e. an ordered list of people that should be interviewed in order to verify their suitability for the new job. In case the gap for a specific role is negative, i.e. indicates a reduction for a specific role, the **RPM,** after exploiting all the possibilities to apply job rotation for the involved people, forwards to **ICA2** only a numeric indication of the people to be moved to other Divisions/sites. Once **RPM** has completed the compilation of the job rotation proposals (respectively, in case of lay off, the numeric indications) for the Lines, the Line Managers start a new cycle of interviews.

In *September*, the new cycle of interviews activates **ICA2** and **IDV**. **ICA2** aims at assessing the competence level of candidates for job rotation in order to select the most suitable people for the unstaffed roles. It is worthwhile noticing that **IDV** is activated during the second cycle of interviews in order to redefine, if necessary, the training and the development plan of the people that are going to rotate their job. Another reason for the activation of **IDV** in September is the completion of **PPE**. The indications of the potential report may determine some changes in the training and in the development plan of the people involved in the potential evaluation.

The **RWD** aims at compensating people with pay raises and level promotions and teams with bonuses. As far as it concerns pay raises and level promotions, the sub-process is activated twice a year: in *June* and in *December*. Bonuses (e.g. patent and "golden block" awards, team rewards) are processed *all along the year*.

Process Application and preliminary results

The process was launched in May 98 and presented to all the staff of the organisation. A copy was also stored in the web. All sub-processes have been applied to 343 individuals in TEI R&D-GPC and good feedback has been received by most of employees during the interviews as well as some improvement proposals.

30 employees have been assessed to evaluate their potential out of 68 candidates.

During December 98 the survey to measure the people satisfaction has been distributed and the results are considered very promising. In fact, comparing them with the results obtained in 1996, the general index increased of 24%.

Detailed results pertaining to some of the survey's modules are listed below:

SURVEY MODULES	**% of increasing**
Work Task	10%
Development opportunities	51%
Information	21%
Culture and climate	33%

Another important parameter to be reported regards the turn over registered in the last 6 months. In fact just 2% of the staff left the company and another 2% asked to be rotated to other divisions.

Conclusions

The actions concerning people management performed during 1998 in TEI R&D-GPC (Ericsson Italy) were of 2 types:

Through the PMDP, people were let to know that more attention was going to be devoted to them, in particular by improving services (in terms of processes) that concerned their inter-work with the company (competence development, carrier path, potential evaluation, etc.).

More importance was given to technical roles and the same dignity as to managerial roles was recognised to them.

These quite long term actions were chosen rather than more direct actions on salaries or bonuses of some individuals because the dissatisfaction was not acute and the turnover was normal (<5%).

The dissatisfaction seemed to stem from the type of inter-work between the individuals and the company rather than from specific claims about the salary.

In the beginning of 1999 a project to implement People-CMM model in TEI R&D-GPC was started and a revision of PMDP was necessary so as to align the process to the new model.

At the end of the revision work some minor changes were introduced, but the overall PMDP structure could stand.

PMDP proved to be a strong infrastructure, robust enough to incorporate the new requirements.

References

[1] B.Curtis, W.E.Hefley, S.Miller People Capability Maturity Model, Software Engineering Institute; September 1995

A Learning Organisation Approach for Process Improvement in the Service Sector

Richard Messnarz, ISCN, Dublin, Ireland

Christine Stöckler, Bernhard Posch, APS, Austria

Gonzalo Velasco, Fueva, Spain

Gearoid O'Suilleabhain, CIT, Ireland

Miklos Biro & Tibor Remszö, Sztaki, Hungary

Introduction

This paper is based on the results from the EU Leonardo da Vinci project Bestregit. Bestregit focussed on general service organizations (general public services, European Union regional information nodes, and regional governmental divisions) which are non-profit, mostly state-funded, non-software and very human centered organizations.

In Bestregit the principles of process improvement were analyzed and tailored for the use in non-software general service industry, and tried out in three half-governmental institutions in Spain, Austria, and Ireland to improve their service capabilities.

The outcome is a framework for process improvement that could be beneficially applied in the general/public service sector to put a structure in place, with goals, and teamwork based processes, aiming at a learning organization architecture.

The paper describes characteristics of general service organizations, the Bestregit methodology, and presents parts of three case studies from the set of experiments tried out in the different countries.

The Approach

The target group in the first place were the technology transfer units (later they called themselves innovation transfer units) which are financed by the regional governments and the EU, and which are responsible for dissemination of programmes, support in the creation of transnational teams, and the multiplication of know how from the EU into the region and vice versa (see Figure 1).

At the beginning of Bestregit there were two different possible approaches to start with:

Trying to invent an ideal architecture of technology transfer and map each technology transfer unit onto this ideal model. (note: the typical assessment and benchmarking approach)
Trying to create a framework of process improvement steps through which each regional technology transfer unit runs and improves their knowledge multiplication ability.

The group for (a) was so ambitious to plan to establish a new technology transfer principle which could be sold to Brussels.
The group for (b) thought much more on a realistic level because:
A small team like Bestregit with regional representatives from a small subset of regions of Europe could not be in the power to influence the European Union level. Regional politics would create troubles, because even if we could manage 1. above, e.g. a regional Austrian unit cannot agree models with Brussels which influence the Austrian state, they would first have to agree on a state level.
There was an expectation of cultural differences so that work models for the same goal might look different in different regions.
Unlike in areas like software industry, electronics industry, etc. there is no international standard which describes the ideal architecture of innovation transfer organisations. There is the ISO 9000 standard with a guideline 9000-4 for service organisations, but this still is a quite general description.

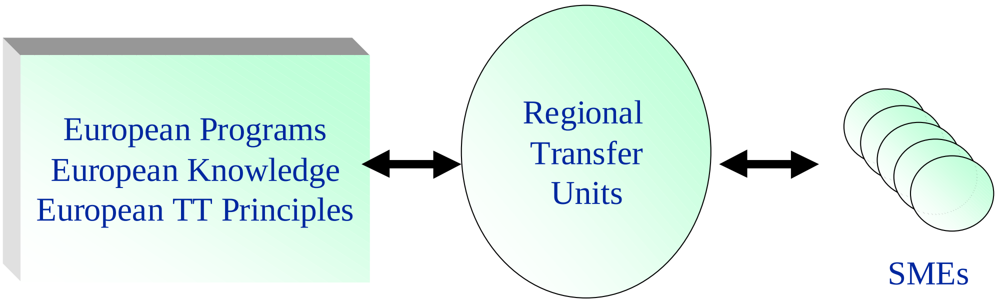


Figure 1 : The Regional Multiplication Nodes

This formed the reason to follow the approach (b) and focus on an improvement of the dissemination and multiplication ability of regional transfer units, thus increasing the multiplication of EU strategies into the regions of Europe.
The project therefore aimed at a synergy between all involved parties. The Eu will have their information and results better promoted, the regional units increase their capability, ad the SMEs will receive a better service through defined teamwork, interfaces, and infrastructure.

What are the Characteristics of a General Service Organisation

The integrated business oriented approach followed in this section was first presented by Dr. Biro in chapter 1 of the book [1] edited by Messnarz, R., Tully, C, 1999, entitled Better Software Practice for Business Benefit – Principles and Experience written in the framework of the EU supported Leonardo PICO (Process Improvement Combined Approach) project.

Special characteristics of the addressed organizations and the BESTREGIT process improvement initiative

The organization we are addressing has the following special characteristics. It is:

- non-profit,

- service oriented,
- its customers are profit oriented enterprises,
- it is sponsored by both public and business organizations.

All of the above characteristics have deep implications for the motivations and approaches of managers in performing and improving their activities.

Service orientation as opposed to manufacturing means that direct contact with the customer is not restricted to the specially trained sales and marketing staff, but is the natural duty of the majority of the personnel. By consequent, technical knowledge is necessary but not sufficient. Special emphasis has to be put on human relationship skills especially when dealing with profit oriented enterprises which are keen on the most efficient use of their resources. This issue is directly addressed by the role based team work approach described in the process improvement guidelines.

'Non-profit' in our case means that the costs of the organization are covered by public sponsors in addition to private ones, which makes the requirement for both high level and impartial service quality of utmost importance.

Advantages and disadvantages of process improvement in non-profit service oriented organizations

One of the most pertinent questions the manager of a non-profit service organization can ask is the following "How can I best satisfy and allow to enhance my customers' and sponsors' expectations using and possibly increasing my available resources?" Managers with financial, operating, production, marketing, human behavioural, or other orientations will give a variety of answers to this question and will arduously argue for their valuable ideas. Here, we will outline a framework integrating and structuring several orientations.

The key concept of the approach is the notion of lever. Levers are means used by a firm to increase its resource generating ability, just as a mechanical lever is used for increasing the force applied to an object. The analogy goes even further. Just as a force can be applied in many different ways to the object resulting in a similar displacement, the use of the different levers can increase the resource generating ability of the firm resulting in similar benefits. Finally, the resources of a non-profit organization are used to increase the assets of the firm and to reward employees.

Let us analyse the ways process improvement can provide leverage to a non-profit service organization from the financial, operating, production, marketing, and human behavioural perspectives.

Financial leverage

Financial leverage means borrowing funds and investing them with a return higher than the cost of the debt. If a company is able to exploit financial leverage, it can make money on funds it does not own. This is by definition a type of leverage a non-profit organization can never exploit. This issue is nonetheless very relevant in our case.

The type of non-profit service organization we are addressing is partly sponsored from public funds which means that there is in fact a public investment whose return can only be accounted for indirectly. This means that the return on investment (ROI) must be realized partly by the public through the intermediary of both the non-profit organization and its client profit oriented organization.

It goes without saying that an exact quantitative return is usually not identifiable in such cases, but a qualitative return statement confirmed by the profit oriented customers and possibly public representatives serves clearly the interests of the non-profit service organization.

We claim that process improvement results in a return definitely making the necessary investment worthwhile only if the addressed organization is fully committed and able to immediately exploit its benefits. Everybody must be aware however that process improvement is not a silver bullet. Commitment and hard work is necessary to obtain the expected results so that the leverages discussed in this section can be taken advantage of.

Operating Leverage

Operating leverage is related to the cost structure, that is the repartition between the fixed costs and variable costs.

Process improvement clearly means an increase in fixed costs, which include training, consulting fees, equipment, improvements in office conditions, etc... However, the question is whether the company is really able to use it for decreasing its variable costs. Measuring the variable costs of a non-profit service organization is not a straightforward issue.

If, due to process improvement, the firm is able to deliver the same quantity and better quality of service using less person months than earlier, then it will have the potential to take advantage of operating leverage.

Learning Leverage

It is an empirical fact that unit costs decline exponentially when experiences are accumulated and the steady reuse of these experiences is well managed by the firm. This is called production leverage in the manufacturing industry, while learning leverage is a better expression in the service sector.

The graph of the unit costs in function of the cumulative quantity of service provided or product produced is called the experience or learning curve which is exponentially decreasing. Its existence is essentially due to economies of scale, learning, improvements, and reuse.

Learning, the accumulation of experiences and the management of their steady reuse is clearly one of the primary objectives of process improvement and it is in the primary focus of the BESTREGIT methodology.

Marketing Leverage

Process improvement, maturity achievement, ISO 9000 certification have an important impact on the perceived capability of the company and on the perceived value of its products or services, which contributes to improved customer satisfaction.

Quality and process improvement are part of a differentiation strategy in which the company delivers and is perceived to deliver a superior product or service. Taking advantage of this marketing leverage towards the public and its customers is clearly the interest of a non-profit service organization.

Human Leverage

It is widely known that employee motivation (empowerment) can be significantly influenced by immaterial means like management styles and organizational structures. Huge individual energies can be released for example in an appropriate teamwork environment where team members are simply given the responsibility to do their jobs as well as they can, instead of exerting close surveillance over them. This means that the same employees can perform more work and even in better quality than otherwise. Nevertheless, attention must be paid at the differences in the collective mental programming of people in different national cultures [57].

The exploitation of human leverage is particularly important in service organizations since they are directly dependent on the enthusiasm of their employees. This issue is largely addressed by the BESTREGIT methodology.

An Overview of the Methodology

The methodology has been adapted from a set of methodologies from the information technology industry and has been field tested in innovation transfer organizations in Austria, Ireland, and Spain.

The methodology works in the following major phases :

Start and First Investment of Resources

Analysis of Current Situation

Goal Analysis

Teamwork Analysis

Experimentation and Measurement

Selection of Experiment

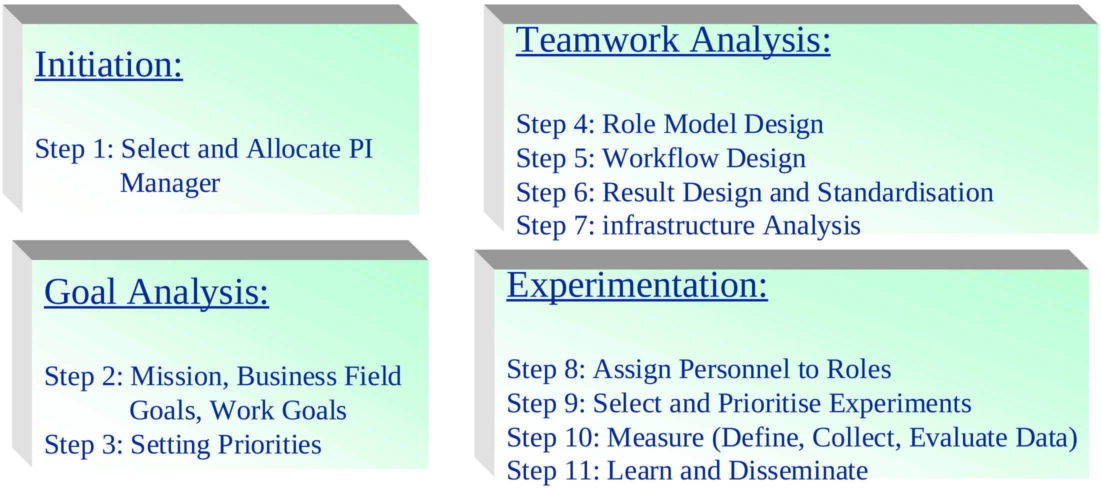
Initiation of Experiment

Experiment Performance

Multiplication of Lessons Learned

The methodology itself helps general service organizations to

- structure the business
- set the goals right
- improve team-work and infrastructure
- build a quantitative feedback loop to learn and change for the future in an objective way.



Initiation:

Step 1: Select and Allocate PI Manager

Teamwork Analysis:

Step 4: Role Model Design
Step 5: Workflow Design
Step 6: Result Design and Standardisation
Step 7: infrastructure Analysis

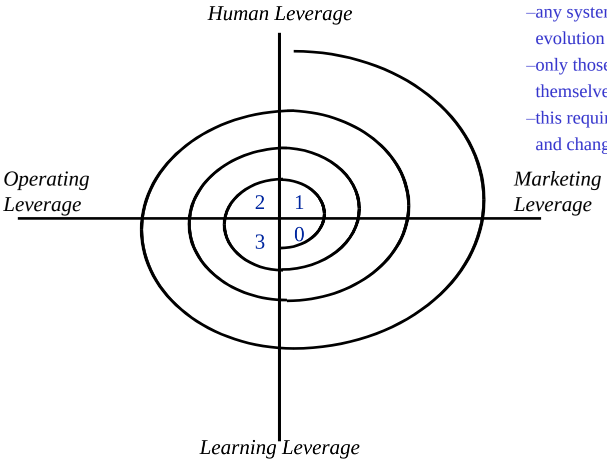
Goal Analysis:

Step 2: Mission, Business Field Goals, Work Goals
Step 3: Setting Priorities

Experimentation:

Step 8: Assign Personnel to Roles
Step 9: Select and Prioritise Experiments
Step 10: Measure (Define, Collect, Evaluate Data)
Step 11: Learn and Disseminate

Figure 2 : The Building Blocks of the Methodology



–any system, process, human being etc. underlies a continuous evolution as a natural principle
–only those stay competitive who are able to adapat themselves ongoingly to the changes caused by the evolution
–this requires continuously learning to manage new situations and change

Every Cycle Includes
- Goal Analysis (L1)
- Teamwork Analysis (L2)
- Experimentation (L3) and Dissemination

Figure 3 : The Learning Framework of the Methodology

From a people point of view (as mentioned before people are the most important resource in service organisations) the following levels of learning are run through:

Level 0 : Initiation (see Figure 3)

At the start:

Our processes are that complex that we do not believe that they can be modelled and shared ? It is just the skills of some heroes fighting for innovation transfer success.

Later: A continuous restart to work on further needs.

Level 1 : Awareness (see Figure 3)

A goal analysis helps to put a structure (with goals) in place for the organisation and makes a common understanding possible.

At this level the visions, goals, and structures can be shared and understood. It is the first time that the hero culture changes into a team with a shared vision. Please note that through a structure (documented and understandable for all) all can contribute and share the goals from that point onwards.

Level 2 : A Team View (see Figure 3)

An information flow and process analysis helps to identify team roles, information flows, work results, and required resources for different work scenarios in the organisation.

At this level people who work in a team start to separate responsibilities, make information flows clear, and agree on work results. This leads to defined team-work structures for different business cases of the service enterprise.

From that point onwards people know their roles, know how to work in a team and how to exchange information to jointly reach a certain service goal.

This level usually ends with an enthusiastic feeling of the people involved. But ….

Level 3 : The Level of Criticism/Feedback (see Figure 3)

No learning environment works without criticism and feedback loops to further build on the goals and the team-work scenarios.

An experiment and try-out shows if the models and goals are right or have to be adapted. This has to be based on measurement of data (objective evaluation).

This usually leads first to a pessimistic phase (after the enthusiastic one) until people realise that the continuous industrial change will always require adaptation and further refinement of the models and goals.

Level 4 : A Change Driven Learning Organisation (see Figure 3)

People understand that change is a natural requirement and that change can be managed (avoiding level 0) as long as there is a structure in place which allows to share goals, work processes, and knowledge.

Change is not a single-person activity it is a change of the shared goal, vision and work force of the entire team following an adapted structure for the organization into the future.

Experiences Gathered per Step

The project created a guideline based on the feedback from practical case studies in regional innovation transfer organisations in Spain, Ireland, and Austria. The guideline is too big to present all underlying work procedures and support tools here.

Below you only find a short list of lessons learned per step

Step 1 : Installation of a Process Improvement Manager (Team)

This improvement manager is responsible for

- performing the improvement steps outlined in this guide-line
- organizing and performing the goal analysis workshop and preparation of a consistent goal tree
- organizing and performing the workshops for team analysis and preparation of consistent team communication, role, and work-scenario models
- presenting and comparing the results with all other partners, units, departments and innovation transfer organizations
- organizing and conducting experiments to try out the improved practices and measure the impact
- evaluating if the goals have been achieved and initiating further changes and refinements

Goal Analysis

Mission. A mission is a strategic goal of the organization. It is usually defined by the director and the board of owners, has a long term view, and is defined in a way that it clearly represents the organization and allows that all current business fields fit into it. Especially in innovation transfer it is usually aligned with some technology transfer political visions.

You can see that a mission is right if (in spite of the rapidly changing requirements) it had to be changed only every 4-5 years.

Business Field Goals. This is a goal that is important to be achieved to satisfy the mission. To reach such a business field goal it is required to create a work force whose task it is to perform projects which contribute to the achievement of that business goal. Such goals usually are defined in cooperation between the director and the department heads (managers) of the organization.

A goal for a certain business field is right if (in spite of the rapidly changing requirements) it had to be changed only every 2-3 years.

Work Specific Goals. This is the most concrete level of a goal. Work specific goals contribute to the achievement of business field goals. Here concrete work performance indicators can be measured and conclusions are made if the project was successful or not.

Such goals usually are defined in cooperation between the department heads (managers) and project leaders of the organisation.

A goal for a certain project is right if it clearly contributes to the business field goal and if it is short term (maximum 1.5 years) and can be measured to have an objective basis to decide about success, failure, and required adaptations.

Goal Tree. A goal tree defines an architecture in which the mission, the business field goals, and the work specific goals are represented. A goal tree (in its ideal structure) should provide forward and backward trace-ability. Forward trace-ability means that it is clear which business fields belong to the mission, and which work specific goals contribute to which business field goal. Backward trace-ability means that quantitative performance indicators are used for measuring the achievement of the work specific goals, and that these indicators help to evaluate a business field indicator, which in turn can be used to measure the mission's success. Backward trace-ability builds the feedback loop into the goal structure, and usually it requires data collection and evaluation.

Step 2: Identification of the Mission, Business Field Goals, and Project Goals

Lessons Learned

- A goal tree with no backward trace-ability is a nice picture but cannot be verified in its operation. Be aware of that. Backward trace-ability means that quantitative performance indicators are used for measuring the achievement of the work specific goals, and that these indicators help to evaluate a business field indicator, which in turn can be used to measure the mission's success.
- Goal trees are a translation of perspectives of different staff levels in an organisation. In fact the process improvement manager helps to create a structure which translates the top manager's view to become understandable for the business field managers, and translates the business field manager's view to become understandable for the staff and project managers. Often it highlights that people work without specific goals that could be tracked. Be aware of this translator role and potential human communication conflicts.
- Use simple metrics and measures as quantitative indicators which can be easily collected and evaluated. Do no invent new types of data for which the organisation must make big efforts to collect and measure.

Step 3: Setting Priorities Before Further Effort Investment

While in Bestregit the three organizations got funding to model all work scenarios, in a real business case this approach would not be applicable. Business and service demands lead to
Limited time
Limited resources
Priorities of the organization.

It means significant effort to run at least one improvement, and to achieve return on investment you have to invest your improvement resources properly and carefully.

Therefore

- Select the business field which is most promising at the moment

- Select within that business field a typical work scenario which highly contributes to the success of the business field
- Give a quantitative rationale for the selection (why, which impact has it now and which is expected)

Lessons Learned

- If you select a business field with a typical work scenario, do not forget the “re-use factor”. If you model a work scenario that is just used once, it is not worth the effort. Only a modeling of those scenarios is most fruitful which can be many times re-used (are used for a long time again and again), shared and multiplied amongst a group of people.
- If you want to select more than one business field and scenario, do not try more than 3 at the same time. Experience showed, that even in big organizations with larger resources more than 3 experiments in parallel got critical.
- If you have a clear organigram usually each department represents a business field. However, experience shows that goal analysis leads to refinements in the organigram.

Teamwork Analysis

Work Scenario. Each organization consists of a set of work scenarios. E.g. Customer handling, service delivery, workshop organization, etc. A work scenario is therefore a description of the best way to conduct a certain business case in the organization.

Work scenarios in Betregit are described with two complementary views:

Role Models. Role centered models base on roles which are played by individuals. One person can play many roles as well as many persons could play just one role. Roles exchange information and work results. This information flow between the roles forms the role model.

Work Flow Models. Work flow models consist of a network of work steps. Work steps produce results that can be used by other work steps. Each work step requires resources (e.g. a certain effort, tools to be used, etc.).

Bestregit uses an integration of both these views:

First role models are analysed and designed. Secondly the role models are transformed into work-flow views. Thirdly, both models are integrated so that a work scenario according to Bestregit can be defined as follows:

A **Work Scenario According to Bestregit**. People are assigned to roles, roles are assigned to activities, activities are part of a network of work-steps, activities produce results, and roles use resources to perform the activities. These relationships are then defined for a certain business case of the organization to have a description of the best way to perform the business case.

Step 4: Identify the Roles and Design a Role Model

Lessons Learned:

- One person can play many roles. One role can be played by many persons. There is no one to one relationship between people and roles !
- One role could be part of different work scenarios.
- Each team-work scenario should not have more than 5 – 7 roles as a maximum.

- Not all information flows between roles should be modelled, otherwise there would be hundreds of arrows. Only those arrows should be included which carry a result (e.g. document, report, etc.) to be exchanged
- Relate to a control specific flow (review and / or test and / or acceptance inputs)
- Typical roles are director, manager (of a business field), team co-ordinator, designer, consultant, customer, partner, etc.

Step 5: Identify the work-steps and create a work-flow

Lessons Learned:

Step 5 can result in the identification of a number of inconsistencies, such as

- A work step to be carried out by a role, although the responsibilities/activities of that role (as a result of workshop 1) do not include this work step. → Refine the role description.
- A work step to be carried out by a role, which failed to be identified in workshop 1. → Include the new role in the role models, and describe it.
- A responsibility/activity defined for a role which should be a work step, but this work step failed to be identified in step 5 → Include the additional work step in the corresponding work flow.

An elegant approach for ensuring consistency is to transform role models into work-flow views. The Bestregit guideline contains a procedure for how to manage this.

Step 6: Identify the Results Produced by the Work-Steps

After performing the previous steps 1 – 5 the organization is understood as a defined network of work steps, performed by roles, with a number of results to be produced in a team. However, so far the results are just a graphical element in a work-flow chart or the name of an arrow in a role model.

The question is "how can we evaluate if a work-flow model or a role model is right"? Here the Bestregit methodology takes into account different perspectives:

Quality. The degree to which a system, component, process, or service meets customer or user needs or expectations. [IEEE-Software Engineering Standards]

Customer´s Quality Perception. A customer would not look into how the processes were used to develop a service or product, he would just evaluate the quality of the delivery. (how his/her expectations are met).

Manager's Quality Perception. A manager who coordinates work and delivers products or services to a customer evaluates the quality of his work on how well structured his work processes are to ensure that he can deliver quality to customers.

Bestregit's Perception of Quality. Quality of an end product / service (on which the customer perception bases) is the sum of the quality of the intermediate results of the work steps in the work flow (what the manager can review and control).

Therefore the Bestregit methodology assumes that

- Those results (documents, reports, administrative storages, leaflets, products, etc.) which contribute to the quality of the end product / service and have been identified in the role and work – flow models should be analyzed and a "best-to-contain" structure proposed with a template.

- A review mechanism (check if the reviewer, quality manager, and improvement manager roles are there !) is to be installed to ensure that the intermediate products are reviewed and compared with the “best-to-contain” structure (if and how they used the template).

Experimentation

Metric. A quantitative measure of the degree to which a system, component, process, or service possesses a given attribute. [IEEE-Software Engineering Standards]

Measurement. The activity of assigning numbers using *a defined counting or evaluation process* (a metric) on the characteristics of a product or activities.

Experiment. An experiment is a practical try-out of the previously established models and results. It must be based on a feedback mechanism that allows to measure the impact of the experiment at the start and after performance of it. Both measures are then used to make conclusions about success/failure of the experiment and required refinements.
An experiment is the first step to create a learning/feedback loop around ideal models established by the managers’ quality perceptions.

Three types of experiments were tried out in Bestregit –

Type 1 - Work Scenario Optimisation

One selected work-flow is chosen for a try-out. The required duration times and efforts for the roles are set at the beginning either based on previous experience or assumptions (if no past data are there). People are assigned to the roles and the work-flow is initiated.
Deviations, problems, and improvement wishes are documented throughout the experiment controlled by the process improvement manager.

This includes

- Longer duration times than expected
- More or less effort than expected
- Work steps not required and additional required work steps
- Buffer times needed (when a work step is long in an wait-status due to inputs from outside)

The experiment results in a refined work model, with adapted duration and effort times. This is like running through a set of learning cycles, and the more cycles you run, the more realistic and professional the processes become.

Type 2 – Infrastructure and Work Process Optimisation

The Bestregit methodology analyzed all required ingredients to build a computer supported work environment (Intranet) in the steps 4 to 6.

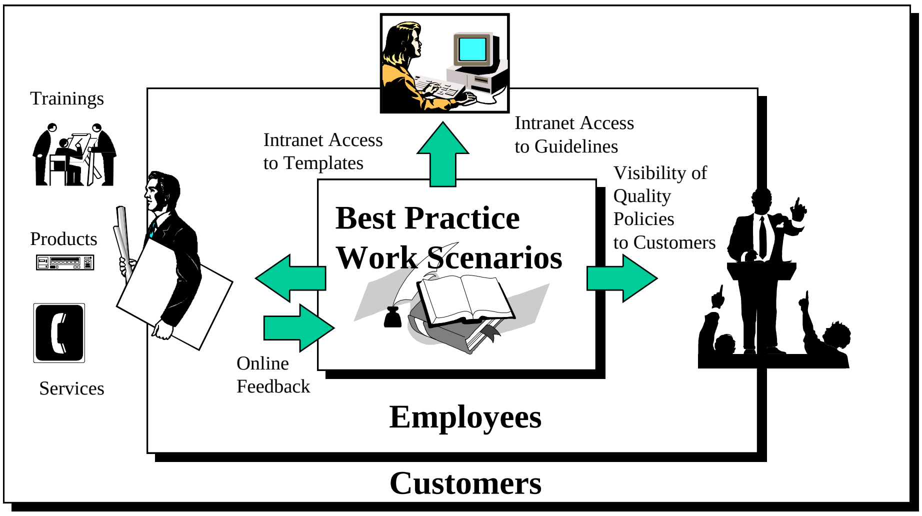

Trainings
Products
Services
Intranet Access to Templates
Intranet Access to Guidelines
Visibility of Quality Policies to Customers
Best Practice Work Scenarios
Online Feedback
Employees
Customers

Figure 4 : Intranet Work Environment Using Bestregit Like Results

Role and work flow models as "best practice work scenarios" can be made visible to all staff
Result templates can be accessed and ensure a common quality in design and documentation throughout the projects. An "index" attribute of the results helps to find the template within the Intranet.
The goal architecture can be made accessible to customers increasing the customer trust in the organization's vision.
For each project (following certain work scenarios) project archives are created producing and archiving results (based on the result templates)
On-line feedback should be installed from the staff to the improvement manager to keep the feedback/learning cycle alive.

A typical measurement at the start would be different satisfaction evaluations:
Evaluation of the staff satisfaction with the work environment and procedures.
Evaluation of the customer satisfaction with the products/services delivered.

With the assumption that after implementation of the infrastructure, and the computer support of best practices, work can be better done, is more complete, traceable and will deliver as promised.

Type 3 - Human Resource Capability Optimisation

In the past skills were largely defined on a single-person level. However, the systems and services in industry become that complex, that sometimes one person would work for her/his life-time or more to complete the task. So team work is growing in its importance and in the new education and skills white paper of the European Union inter-personal and communication skills are emphasised. This is also the focus of the Bestregit methodology which creates frameworks (role models) that allow people to identify themselves with roles, know their interfaces to other team members, and through a feedback cycle can build on the team work structure.

The assumption is that if you try out a Bestregit role model in staff training and integration that the time to be integrated and effective for organisations decreases. The staff integration time is reflected by the effort of the person which introduces the person to the environment. The less additional effort is needed to become a new effective employee, the smoother the integration works. This tutoring effort can be used as an indicator.

Lessons Learned

Step 8: Assign your Personnel to the Roles

Step 9: Select and Prioritise Experiments

Step 10: Measure (Define, Collect, Evaluate Data)

Step 11: Learn and Disseminate

Here we describe some example results from the field test organisations. Each field test report is prepared as a case study and comprises about 60 pages. Thus we only can extract small parts for this paper.

Example E: Spanish Organisation – Sample Types 2 & 3 Experiments

The Spanish filed test partner is a non for making profit organisation created by the Valladolid Chamber of Commerce and Valladolid University in 1986 in order to link interests coming from both institutions.
They have a large background in the management of Regional, National and European projects. They are mainly specialised in Training and Human Resources Projects, and in Innovation projects and Technology Transfer.

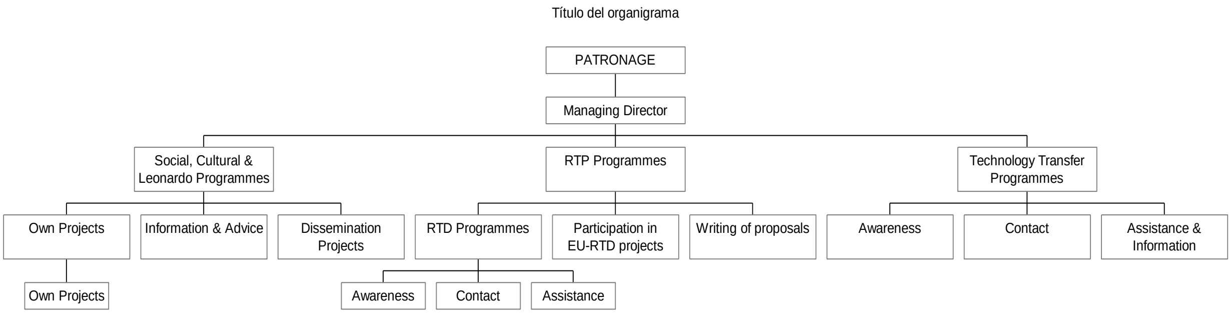
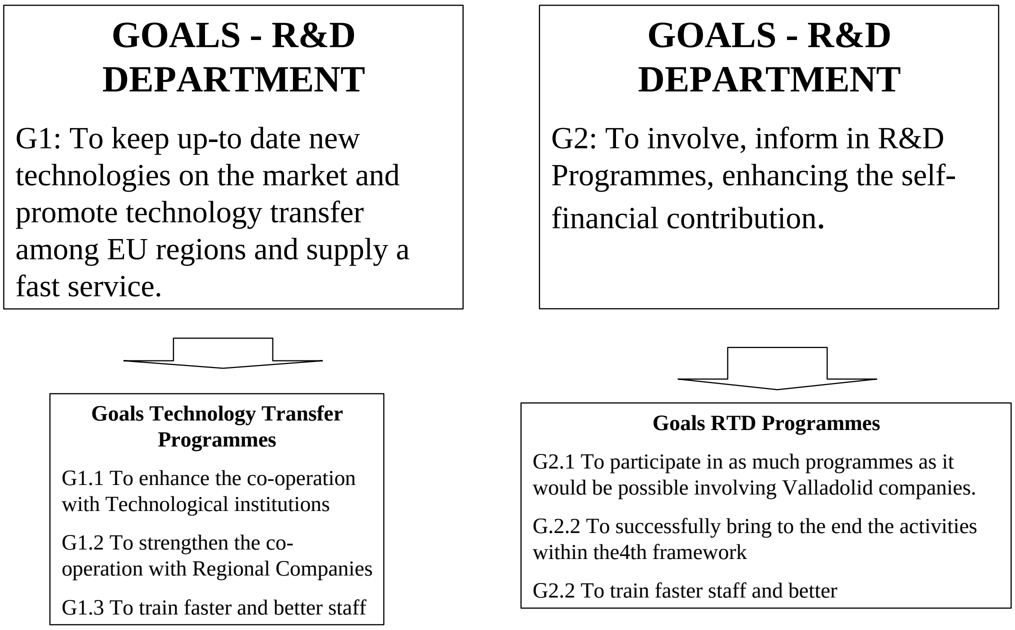


Figure 5 : The Spanish Partner Structure and Identified Work scenarios



GOALS - R&D DEPARTMENT

G1: To keep up-to date new technologies on the market and promote technology transfer among EU regions and supply a fast service.

GOALS - R&D DEPARTMENT

G2: To involve, inform in R&D Programmes, enhancing the self-financial contribution.

Goals Technology Transfer Programmes

G1.1 To enhance the co-operation with Technological institutions

G1.2 To strengthen the co-operation with Regional Companies

G1.3 To train faster and better staff

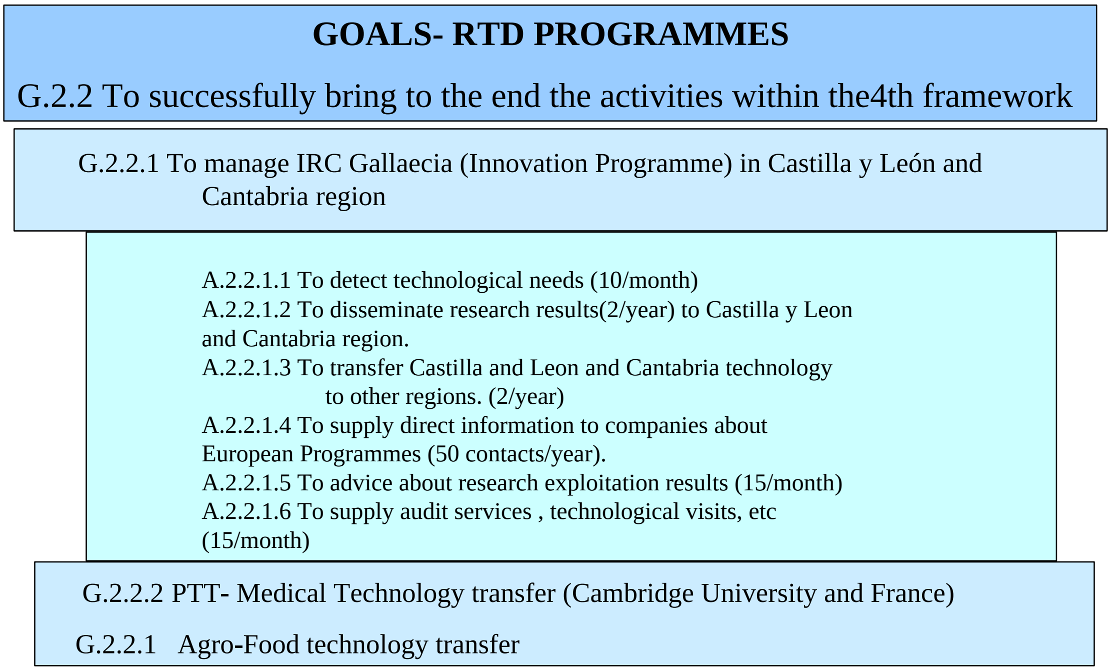
Goals RTD Programmes

G2.1 To participate in as much programmes as it would be possible involving Valladolid companies.

G.2.2 To successfully bring to the end the activities within the4th framework

G2.2 To train faster staff and better

Figure 6 : The Spanish Partner Top level Goals



GOALS- RTD PROGRAMMES

G.2.2 To successfully bring to the end the activities within the4th framework

G.2.2.1 To manage IRC Gallaecia (Innovation Programme) in Castilla y León and Cantabria region

A.2.2.1.1 To detect technological needs (10/month)
A.2.2.1.2 To disseminate research results(2/year) to Castilla y Leon and Cantabria region.
A.2.2.1.3 To transfer Castilla and Leon and Cantabria technology to other regions. (2/year)
A.2.2.1.4 To supply direct information to companies about European Programmes (50 contacts/year).
A.2.2.1.5 To advice about research exploitation results (15/month)
A.2.2.1.6 To supply audit services , technological visits, etc (15/month)

G.2.2.2 PTT- Medical Technology transfer (Cambridge University and France)

G.2.2.1 Agro-Food technology transfer

Figure 7 : Goals and Measurements

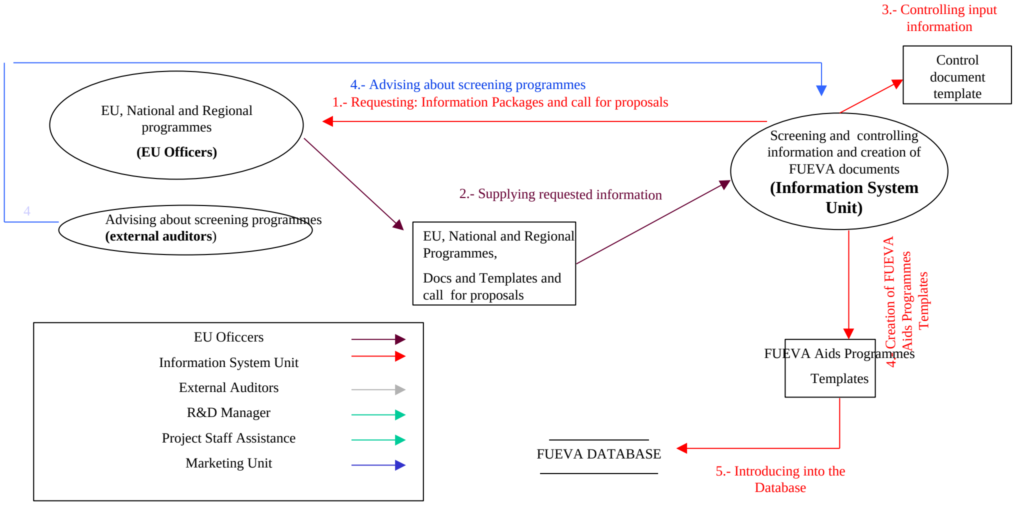

3.- Controlling input information
Control document template
4.- Advising about screening programmes
1.- Requesting: Information Packages and call for proposals
EU, National and Regional programmes (EU Officers)
Screening and controlling information and creation of FUEVA documents (Information System Unit)
2.- Supplying requested information
Advising about screening programmes (external auditors)
4
EU, National and Regional Programmes, Docs and Templates and call for proposals
4.-Creation of FUEVA Aids Programmes Templates
FUEVA Aids Programmes Templates
EU Oficcers
Information System Unit
External Auditors
R&D Manager
Project Staff Assistance
Marketing Unit
FUEVA DATABASE
5.- Introducing into the Database

Figure 7 : A Small Part A of the The R&D Management Role Model

The experiment focussed on the design, development and implementation of a documentation system, which allows everybody to manage information/documentation of the rest of colleagues from the same department without having problems to find any document.
To measure, in quantitative and qualitative way, all improvements made through the comparison between the current and future situation.

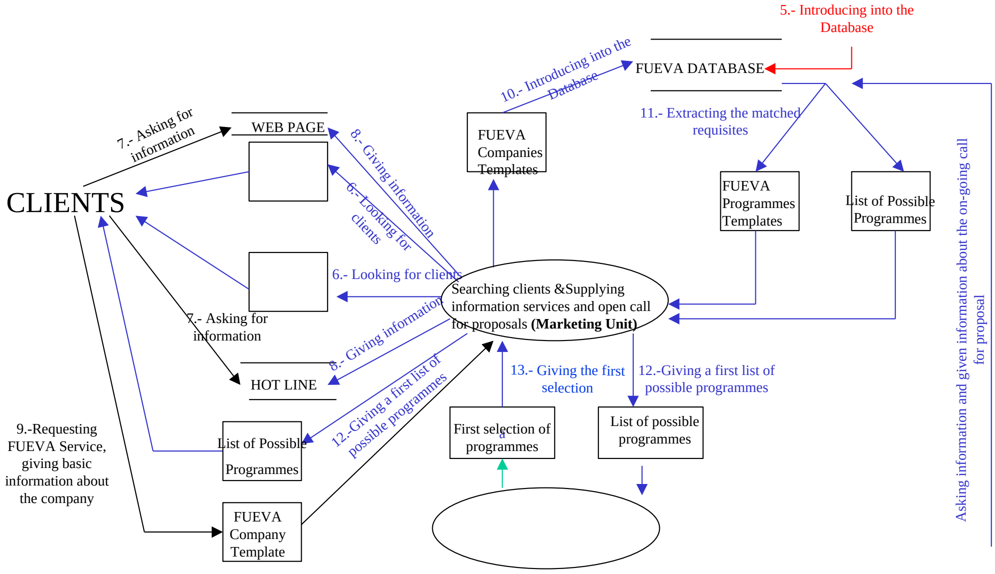

5.- Introducing into the Database
10.- Introducing into the Database
FUEVA DATABASE
11.- Extracting the matched requisites
7.- Asking for information
WEB PAGE
8.- Giving information
6.- Looking for clients
FUEVA Companies Templates
FUEVA Programmes Templates
List of Possible Programmes
CLIENTS
Asking information and given information about the on-going call for proposal
6.- Looking for clients
Searching clients &Supplying information services and open call for proposals (Marketing Unit)
7.- Asking for information
8.- Giving information
HOT LINE
13.- Giving the first selection
12.-Giving a first list of possible programmes
12.-Giving a first list of possible programmes
9.-Requesting FUEVA Service, giving basic information about the company
List of Possible Programmes
First selection of programmes
List of possible programmes
FUEVA Company Template

Figure 8 : A Small Part B of the The R&D Management Role Model

The Bestregit analysis results included all ingredients to build such a system

- Clear goals
- Clear roles
- Clear work flows
- Agreed structure and content of materials and results (including templates)

So the Spanish partner built up an indexing system through a database which stores information and resource allocators of project results.

Results:

- A questionnaire analysis was done with all staff before and after the experiment and the results showed that while before it above 50% said that they cannot find materials if the responsible person is not available, after the experiment all confirmed that they can find now the materials.
- The role models (clear responsibility and team interfaces) led to a large reduction of the required tutoring effort for new staff to be integrated.

References

[1] Messnarz R. Tully C., Better Software Practice for Business Benefit – Principles and Experience, IEEE Computer Society Press, Brussels, Washington, Tokio, 1999, ISBN : 0-7695-0049-8.

Goal Analysis

[2] Basili, V., Rombach, H.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, Vol 14, Number 6, June 1988.

[3] Card D.: Understanding process improvement, *IEEE software*, July 1991.

[4] Debou C.: ami a new paradigm for software process improvement, In: *Proceedings of the first ISCN Seminar*, Dublin, May 1994

[5] Debou C., M. Kuntzmann-Combelles A: From Business Goals to Improvement Planning,: Practical Use of ami In: *Proceedings of the SPI96/ISCN96 conference*, Brighton, December 1996

[6] Debou C., Stainer S.: Improving the maintenance process: a quantitative approach, In: *Proceedings of the 6th international conference on software engineering and its application*, Paris, Nov 1993.

[7] Debou C., Fuchs N., Haux M.: ami: a Taylorable Framework for Software Process Improvement, In: *Proceedings of the second ISCN Seminar*, Vienna, September 1995

[8] Debou C., Kuntzmann-Combelles A., Rowe A.: A quantitative approach to software process, In: *Proceedings of the 2nd international symposium on software metrics*, London, Oct 1994.

[9] Dion R.: Process Improvement and the corporate balance sheet, IEEE Software, July 1993.

[10] Debou C., Lipták J., Shippers H.: Decision making for software process improvement: a quantitative approach, In: *Proceedings of the 2nd international conference on "achieving quality in software" ACQUIS 93*, Venice (Italy), pp 363-377, Oct 1993.

Also In: *The Journal of Systems and Software*, 1994; 26:43-52, Elsevier Science Inc.

[11] Esprit II 5494 ami:ami Handbook, Published version, March 1992.

[12] Mc Garry F. and al.: Software process Improvement in the NASA Software Engineering Laboratory, CMU/SEI-94-TR22, December 1994.

[13] Kuntzmann-Combelles A., from assessment to improvement actions: compared experiences with CMM and SPICE, In: *Proceedings of the 5th European conference on software quality*, Dublin, Sept 1996.

[14] Perez I., P. Ferrer, A. Fernandez: Application of Metrics in Industry In: *Proceedings of the 3rd European conference on software quality*, Madrid, November 1992.

[15] Pulford K., Kuntzmann-Combelles A., Shirlaw S.: A Quantitative Approach to Software management, ISBN 0201877465, Addison Wesley, 1995

[16] Rombach, D.: New Institute for Applied Software Engineering Research, In: *Software Process Newsletter*, No 7, IEEE Computer Society TCSE, Fall 1996

[17] SEL, NASA Goddard Space Flight Center, Software Engineering Laboratory Relationships, models and management rules, SEL-91-001, February 1991

Process Definition and Teamwork Analysis

[18] Chroust G., Computer Integrated Work Management, in (ed.) Mittermeir R., *Shifting Paradigms in Software Engineering*, pp. 4 -13, Springer Verlag, Wien, New York, Sept. 1992

[19] Curtis B., M. Kellner, J. Over: Process modelling, In: *Communication of the ACM*, September 92, vol 35, No 9.

[20] ESA Board for Standardisation and Control, *ESA PSS 05 Software Engineering Standards*, European Space Agency, Paris, 1991

[21] German Interior Ministry, German V-Model, Bonn, August 1992

[22] Haase V., Messnarz R., Koch G., Kugler H., Decrinis P: Bootstrap: Fine-Tuning Process Assessment, In: *IEEE Software* pp25-35, July 1994

[23] Haase V., Messnarz R., Cachia R.M., Software Process Improvement by Measurement, in (ed.) Mittermeir R., *Shifting Paradigms in Software Engineering*, pp. 32 - 41, Springer Verlag, Wien, New York, Sept. 1992

[24] Haase V., Messnarz R., *A Survey Study on Approaches in Technology Transfer, Software Management and Organisation*, Report 305, Institutes for Information Processing Graz, June 1991

[25] J. Herbsleb, A. Carleton, J. Rozum, J. Siegel, and D. Zubrow: „Benefits of CMM- based Software Process Improvement: Initial Results“. Technical Report, CMU-SEI-94-TR-13, Software Engineering Institute, 1994.

[26] Humphrey W.: Managing the Software Process, Addison-Wesley, Reading, Mass., 1989.

[27] Humphrey W.S., Sweet W.L.: A Method for Assessing the Software Engineering Capability of Contractors, Software Engineering Institute, Sept 1987

[28] ISO 9000-3. Quality management and quality assurance standards. International Standard. Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software. ISO (1990).

[29] ISO 9001. Quality Systems. Model for Quality Assurance in Design/Development, Production, Installation and Servicing. International Organisation for Standardisation, Geneva (1987)

[30] ISO 9126, Information Technology - *Software Product Evaluation* - Quality Characteristics and Guidelines for Their Use, 1991

[31] ISO/IEC 12207, *Information technology - Software life cycle processes*, first edition Aug. 95.

[32] Messnarz R., Kugler H.J., BOOTSTRAP and ISO 9000: From the Software Process to Software Quality, in: *Proceedings of the APSEC´94 Conference*, Comput. Soc. Press of the IEEE, Tokyo, Japan 1994

[33] Messnarz R., Practical Experience with the Establishment of Improvement Plans, in: *Proceedings of the ISCN´96/SP'96 Conference on Practical Improvement of Software Processes and Products*, ISCN LTD, Brighton, UK, 1996

[34] Messnarz R., Scherzer H., The Evolution of a Quantitative Process Analysis - the BOOTSTRAP - Approach, in: (eds.) G. Chroust, P. Doucek, Interdisciplinary Informational Management Talks, Oldenbourg, Vienna, Munich, 1995

[35] Messnarz R., Stubenrauch R., Melcher M., Bernhard R., Network based teamwork and Quality Assurance (NQA), in: Proceedings of the 6th European Conference on Software Quality, Vienna, April 1999

[36] Paulk M. C., Curtis B., Chrissis M. B.: Capability Maturity Model for Software, version 1.1, CMU/SEI-93-TR-24, February 1993.

[37] Paulk M. C., Weber C. V., Garcia S. M., Chrissis M : Key practices of the Capability Maturity Model, version 1.1, CMU/SEI-93-TR-25, February 1993.

[38] Axel Völker, Sortware Process Assessments at Siemens as a Basis for Process Improvement in Industry, Proceedings of the ISCN'94 Conference, ISCN Ltd., Dublin, Ireland

Measurement and Experimentation

[39] Boegh J., SCOPE - A Guide for Software Product Quality Evaluation, in: *Proceedings of the ISCN´94 Conference on Practical Improvement of Software Processes and Products*, ISCN Ltd., Dublin, Ireland, 1994

[40] Briand L., El Eman K., Melo W.: AINSI: an Inductive Method for Software Process Improvement: concrete Steps and Guidelines, In: *Proceedings of the second ISCN Seminar*, Vienna, September 1995

[41] Briand L., Differding C., Rombach D.: Practical Guidelines for Measurement-based Process Improvement, Technical Report of the International Software Engineering Network, ISERN-96-05, 1996

[42] L.C. Briand, C.M. Differding, H.D. Rombach, Practical guidelines for Measurement Based Process Improvement, Proceeding of SP-ISCN 96 Conference, Brighton, December 1996

[43] Brooks F.P., The Mythical Man-Month, in: Datamation, Addison Wesley Publishing Company, Massachusetts, December 1973

[44] DeMarco T., *Controlling Software Projects*, Yourdon Press Computing Series, Prentice Hall, Englewood Cliffs, London, Sydney, Tokyo 1982

[45] R.B. Grady, D.L. Caswell, Software metrics: establishing a company-wide program, Prentice-Hall, 1987

[46] R.B. Grady, Practical software metrics for project management and process improvement, Prentice-Hall, 1992

[47] G. Bazzana, P. Caliman, D. Gandini, R. Lancellotti, P. Marino, Software management-by-metrics: experiences in Italy, Invited paper - Proceedings of CSR 10th Annual Conference, Amsterdam, October 1993.

[48] G. Bazzana, R. Brigliadori, O. Andersen, T. Jokela, ISO 9000 and ISO 9126: friends or foes, Proceedings of IEEE Software Engineering Standards Symposium, Brighton, September 1993

[49] B. Hetzel , The sorry state of the art of software measurement, in: N. Fenton, R. Whitty, Y. Iizuka, Software Quality Assurance and Measurement - A worldwide perspective, Thomson Computer Press, 1995

[50] Mehner T., Siemens Process Improvement Approach. In Practical improvement of software processes and products: Proceedings of the ESI–ISCN 95 Conference on Measurement and Training Based Process Improvement, Vienna, September 1995.

[51] METKIT Consortium and BRAMEUR Ltd., METKIT Industrial Package, 1994

[52] Pyramid Consortium, Quantitative management: get a grip on software!, Technical Reference: EP-5425 Y 91100-4, December 1991

[53] E. Trodd, A Minimum Set of Metrics for Effective Process Management, Proceeding of SP-ISCN 96 Conference, Brighton, December 1996

[54] E.F. Weller, Using Metrics to manage Software Projects, IEEE Computer, Vol. 27, No. 9, September 1994

[55] G. Stark, R.C. Durst, C.W. Vowell, Using metrics in management decision making, IEEE Computer, Vol. 27, No. 9, September 1994

Business and Leveraging Models

[56] Capers Jones, The Pragmatics of Software Process Improvement. Software Process Newsletter. No.5, Winter 1996, pp.1-4.

[57] Geert Hofstede, Motivation, Leadership, and Organization: Do American Theories Apply Abroad? Organizational Dynamics. Summer 1980, pp.42-63.

[58] Herbsleb,J; Carleton,A; Rozum,J; Siegel,J; Zubrow,D. Benefits of CMM-Based Software Process Improvement: Initial Results. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-94-TR-13.

Process Improvement: The Societal Iceberg

Kerstin V. Siakas [2,1] and Elli Georgiadou[1]

[1] University of North London
School of Informatics and Multimedia Technology
2-16 Eden Grove, London, N7 8DB
Tel.: + 44 171 753 3142, Fax: + 44 171 753 7009
E-mail: k.siakas@unl.ac.uk, e.georgiadou@unl.ac.uk

[2] Technological Educational Institution of Thessaloniki,
Department of Informatics,
P.O. Box 14561 ,GR-54101 Thessaloniki, Greece
Tel.: +30 31 791296, Fax: +30 31 799152
E-mail:siaka@it.teithe.gr

Abstract

Information Systems (IS) are integrated systems for providing information to support operations, processes, management analysis and decision-making functions in an organisation [1]. The IS area is characterised by rapid technical change and innovation. In recent years there has been a shift in IS from technological to managerial and organisational issues. This shift has led to increased interest in how environment and innovation interact. In software quality improvement efforts there has also been a shift from technical to managerial, organisational and people issues concentrating on the process rather than on the product. It is widely accepted that emphasis has to be placed on process quality as a means of achieving product quality. In this paper, we investigate the characteristics of IS from the managerial and social viewpoint identifying the societal elements, their interacting and their effects on information.

1. Introduction

IS are socio-technical systems [2] of which information technology is one aspect. They can be thought of as integrating an infrastructure and the various systems, which make use of that infrastructure [3]. IS are meaningful only when they are considered within a context. The distinction between a software system and an information system is that software is limited to the development process of a software system, while an information system is seen to be the organisational context in which software is used [4]. If we accept this distinction software quality means development process quality leaving out the usage of software, while IS quality will stress product quality assessed by the usage of software in an organisational context. Due to the multidisciplinary character of IS a discussion about the necessity of a societal viewpoint in these days of globalisation of the software market, virtual global enterprises and cross-cultural teams follows with emphasis on software quality and process improvement.

2. The Societal Perspective

The globalization of industries has caused the contextual boundaries of IS research and practices to include the societal context. IS departments face many challenges in today's rapidly changing and highly competitive global environment. The need for compatible standards and procedures within the global network is obvious. Greater attention to national factors, like national culture, economic structure, political and legal environment and nations' infrastructures, is being given in recent years. The study of societal context enables researchers and practitioners to improve their understanding of the impact information technology on society as well as the influence society has on the development and use of information technology. Aspects of societal environment have been found to be important especially in transnational context [5, 6]. Understanding these aspects enables IT managers at multinational organisations to operate more appropriately in countries other than their own. In figure 1 the IS Research Domain is shown.

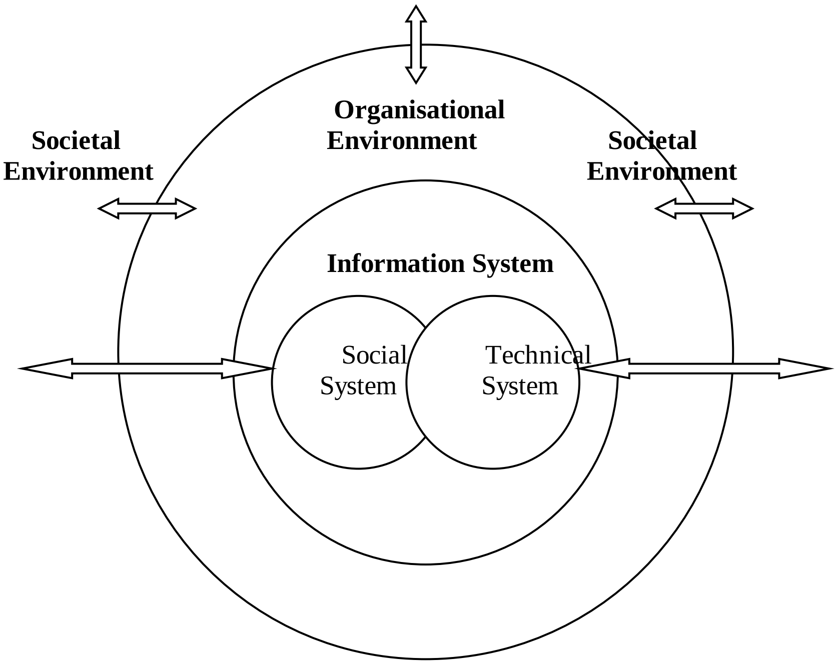


Figure 1. The IS Research Domain adapted from Traut et al.

There are two views on managing information technology (IT) in a global context. One view proposes that managing IT in a global context is largely the same as managing IT in a domestic context. The other view proposes that there are differences depending on cultural aspects, different business and legal environments, different languages and varying technology availability [7]. Some researchers have found that in professions like software engineering the professionals will converge and become more similar to one another because of the universal technology and the consequent creation of jobs, education and training influencing not only skills but also attitudes [8,9,10,11]. On the other hand the successful information system, the evaluation and the maintenance are dependent on user acceptance. There is evidence of national cultural differences and the influence of national culture on organisational culture should be taken seriously. Different researchers have identified how cultures vary. The most notable is Hofstede [12]. Culture is according to Hofstede's definition: "the collective programming of human mind that distinguishes the members of one human group from those of another". His research was carried out in 50 different countries and resulted in a position on four dimensions for each country. The consideration of cultural differences and their effects on IS construction and use is an extremely important issue in the research of international IS. The authors believe that an information system can be successful only when organisational culture is taken into consideration and, if the information system is developed or used in a global context then, the national culture has to be taken into consideration as well.

3. Quality of IS

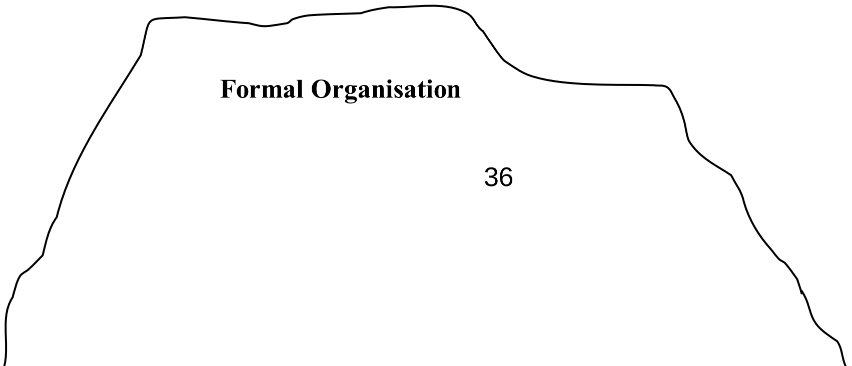
IS Quality includes the requirement of the business organisation, the users and the IT personnel [13]. IS quality is divided into Business Quality, IS Use Quality and IS Work Quality. Business Quality measures the profitability of software investments from the whole organisation's point of view. IS Use Quality is in the interest of all system users, who can be on managerial or operational level, internal or external. IS Work Quality covers the performance level of managing, developing, maintaining and operating IS. Software Quality is usually limited to the development of Software System, while Informations System Quality is seen in the organisational context, where the use of Software is stressed [4].

Because of the underachievement of IS, the over-emphasis on technology has in recent years shifted to take human, organisational and social contexts more into consideration [14]. The quality of IS in general, and software systems in particular, derives from the Total Quality Management (TQM) philosophy, which emphasises customer satisfaction, organisational change and continuous process improvement. Different process oriented quality models, like ISO9001 and process maturity evaluation models also called process capability models, like CMM, Bootstrap and SPICE (ISO-15504) have been developed. Process capability is defined as ability of a process to achieve a required goal [15]. It measures how well a process is managed to achieve its purpose and the organisation's objectives. These models emphasise the implementation of a managed and controlled development process, as well as a process for services necessary to support both the development and use of software. The focus in all these models is on the assessment of the overall technical capability of an organisation. [15]. In 1998 SPICE became the ISO software process assessment standard called ISO15504. The ISO15504-standard combines different methodologies and it is linked with the ISO12207 standard, which provides a framework for software processes. People Capability Maturity Model (P-CMM) [16,17] is an attempt to consider people-issues. P-CMM focuses on three interrelated components namely people, process and technology. The motivation for P-CMM is to improve the ability of software organisations to attract, develop, motivate, organise and retain the talent needed to continuously improve software development capability.

Software is in the heart of most modern businesses. Business success depends on the quality, the cost and the timeliness of the software they use. In order to have a systematic approach to software quality, a software quality management system should be introduced according to the needs of the organisation, and knowledge should be effectively and explicitly managed. The Total Quality movement emphasises that better knowledge of the process will lead to increased productivity as well as higher product quality. Total quality is necessary but not sufficient in today's global economy. New organisations, which go beyond total quality and focus on world competitiveness and future success, are beginning to emerge. These are referred to as learning organisations. They use a knowledge-based approach that focuses on predicting and creatively solving problems by analysing the root causes of problems the first time they appear. The problems are prevented from recurrence by learning how to learn [18].

4. Organisational change

The market driven reason for implementing a software quality management system will require an organisational and cultural change in the organisation. The iceberg metaphor shown in figure 2 can be used to depict the contrasting aspects of organisational life.



- Goals and strategy
- Structure, standards and procedures
- Products and services
- Management and Financial resources

Informal Organisation

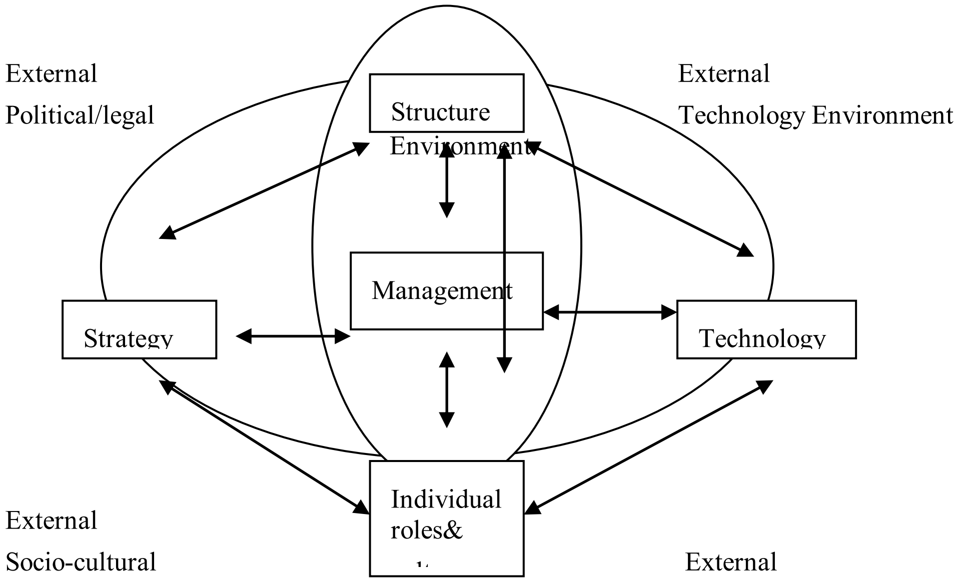
- Values, attitudes and beliefs
- Leadership style and behaviour
- Organisational culture and norms of behaviour
- Power, politics and conflicts
- Informal groupings

Figure 2. The organisational iceberg [20]

The visible part of the iceberg, shows the formal aspects of an organisation while the informal aspects of an organisation hide under water. The informal part is the greater part of the organisational iceberg and will act to help or hinder an organisational process of change. It often leads to resistance to the change process.

The view of organisations existing as systems of interrelated elements operating in multi-dimensional environments is becoming widely accepted. The mnemonics PEST [21] and STEP [22] for example refer both to the Political, Economic, Technological and Socio-cultural factors that influence organisations in their structures, strategies, management process and means of operating including technology and individuals [23].

Figure 3 shows the relationships of the different factors that have to be taken into consideration when implementing a change strategy.



Environment

Economic
Environment

Figure 3 Adapting to change [24]

In order to overcome resistance to process change the software process, the business process, organisational culture and the technology must be understood and managed. The Organisational Development (OD) approach, which is an umbrella term for a set of values and assumptions about organisations and the people within them, together with concepts and techniques, is thought to be useful for long-term organisation-wide change [22]. The OD approach cares about people and believes that people at all levels throughout an organisation are individually and collectively both drivers and engines of change. OD is a process by which behavioural knowledge and practices are used to help organisations achieve greater effectiveness, productivity, and improved product and service quality. The focus is on the process and on improving the organisation's ability to assess and to solve its own problems. It aims to improve the total system, the organisation and its parts in the context of the larger environment that impacts upon them [24]. The success of an OD approach lies in the capabilities of those who act as change agents or champions. A characteristic of an OD process is that it has recognisable phases with activities that help the organisation to move through these phases. In figure 4 basic assumptions of OD as a model for change are shown.

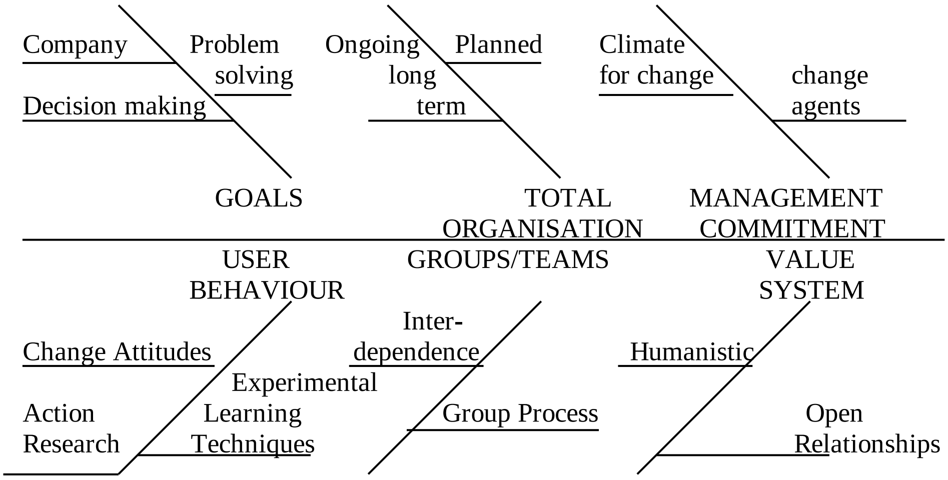


Figure 4 Basic assumptions of OD adapted from Senior [22]

The OD approach to change is entirely in line with the Total Quality Management approach, according to the entire organisation, management commitment and the long-term perspective, extending the approach by regarding people in general as social beings, who form formal and informal groupings as a part of the organisation's functioning. For a group to be effective, all members should share in problem solving and working to satisfy both the task and group members' needs [22].

4. Attitudes for changes in process improvement

To enable the improvement of the software process more attention should be given to individuals in the organisation. The success or the failure of organisational change depends to a high degree on executive open-mindedness toward change and understanding of how individuals and groups function [25]. This is especially important in the increasing globalisation of markets and businesses. Many lessons have to be learned from social and management sciences. Social Science Research involves investigating all aspects of human activity and inter-activity Contemporary Social Research is according to Orlikowski [26] a large range of research perspectives that operates concurrently. Such perspectives are the disciplines concerned with human phenomena such as anthropology, psychology, sociology and their applied fields of administrative science, education, industrial psychology and industrial sociology. Land [27] argues that IS essentially are social systems of which information technology is one aspect and that the study of IS is a multidisciplinary effort.

4.1 Cultural values and Software Quality Management

The globalising trend in recent years has resulted in more cross-national studies. Being a global organisation implies having a universal culture. For the past few decades there has been an important debate about convergence or divergence of work values. International organisations have tried to understand the diverse value system of their multinational structure. The objectives of the multinational organisations is to create a universal culture in the whole organisation and to integrate multi-domestic operations with individuals who hold opposed work related values [28].

The authors are investigating how culture influences the implementation of Software Quality Management Systems, aiming to develop a model, which will take culture into consideration for successful implementation of Software Quality Management Systems. There is evidence that national culture influences management practices and multinational enterprises need to adapt to the national cultures in which they operate in order to achieve high business performance [12,31]. If a multinational organisation is going to be a truly global organisation the diverse individual work values must converge and be integrated into common set of values to create a universal corporate culture [28].

Results from a pilot study carried out in a large Scandinavian multinational organisation, developing software for own use was reported in the Software Quality Management Conference (SQM) in Seville in 1995 [28]. Another study was carried out in three Greek software organisations. The results were reported in the 5th Software Quality Conference in Dundee 1996 [29].

The insights gained from these initial studies helped in reformulating the main hypothesis in this research.

The hypothesis is: *'Cultural factors intervene in the successful application of Software Quality Management Systems'*.

Two cultural variables are identified, namely national and organisational culture**.** Two variables of successful application of Quality Management Systems are identified, namely:

1. The existence of quality oriented management procedures, similar to the procedures identified in Capability Models;

2. The awareness of quality issues amongst the workforce.

The main hypothesis can thus be broken down into four sub-hypothesis:

1. National Culture affects take up of Software Quality Management;

2. National Culture influences awareness of quality issues;

3. Organisational Culture affects take up of Software Quality Management;
4. Organisational Culture influences awareness of quality issues.

In these two pilot studies the CMM self assessment was used to assess the maturity level of the organisations instead of the awareness of quality issues amongst the workforce. Cultural issues were measured and the result was analysed. As a result of these two pilot studies the authors realized that in terms of cultural issues the take up of Software Quality Mangement instead of the quality maturity level is crucial for successful implementation. All proceduers for getting a high score can be in place, but if the workforce does not support them the implementation will not be successful. National Culture will be measured using Hofstede's four dimensions [22]:

1. *Power Distance* :Power Distance Index (PDI) indicates the extent to which a society accepts the fact that power in institutions and organisations is distributed unequally among individuals. In small PDI countries subordinates and superiors consider each other as existentially equal and decentralisation is popular, while large PDI countries subscribe to authority of bosses and centralisation.

2. *Collectivism / Individualism:*Individualism indicates the extent to which a society is a loose social framework in which people are supposed to take care only of themselves and their immediate families. Collectivism is a tight social framework in which people distinguish between in-groups and out-groups and expect their in-group to look after them. In individualist countries people are supposed to take care of themselves and remain emotionally independent from the group. The dominant motivation is self-interest. In collective societies the concern is for the group. Individuals define their identity by relationships to others and group belonging.

3. *Femininity / Masculinity*: Masculinity indicates the extent to which the dominant values in a society tend toward assertiveness and the acquisition of things. In masculine cultures importance is placed on assertiveness, competitiveness and materialism in the form of earnings and advancement, promotions and big bonuses. Femininity indicates the concern for people and the quality of life. In feminine cultures the concern is for quality of relationships and the work of life, nurturing and social well being.

4. *Uncertainty Avoidance*: Uncertainty Avoidance indicates the extent to which a society feels threatened by ambiguous situations and tries to avoid them by providing rules, believing in absolute truths, and refusing to tolerate deviance. In weak uncertainty avoidance countries anxiety levels are relatively low. Aggression and emotions are not supposed to be shown and people seem to be quiet, easy-going, indolent, controlled and lazy while in high uncertainty countries people seems to be busy, fidgety, emotional, aggressive and active.

All the four dimensions are a continuum between two extremes and only very few national cultures, if any, are wholly at one or the other extreme.

The Research Method in the research is a contemporary comparative multimethod using both quantitative and qualitative research methods. The quantitative investigation will be a survey collecting hard data by using a postal questionnaire. The results from the questionnaire will be analysed using sophisticated statistical methods. Subsequently, a qualitative method in the form of case studies will be performed in order to address different aspects of the research problem, to confirm the findings from the questionnaire and to test the hypothesis.

Depending on the findings a conceptual framework will be developed, which is likely to optimise quality and management initiatives in different cultural and organisational settings. This model will couple cultural and organisational aspects with technical requirements of Software Quality Management Systems to ensure successful implementation.

6. Conclusion

By using the Iceberg metaphor we explored the role of informal aspects on the formal aspects of organisations. The globalisation of the software market, virtual global enterprises and cross-cultural teams has caused the contextual boundaries of IS research and practices to include the societal context. We believe that the underachievement of IS depends mainly on the over-emphasis on technology. In recent years emphasis has shifted to take into consideration human, organisational and social-cultural contexts. Quality issues and resistance to change, in particular the degree of resistance influenced by culture were discussed. Many lessons have still to be learned from other sciences due to the great range of research perspectives and the rapid changes in the field. Organisational Development was proposed as an alternative way of empowering capabilities of those who act as change agents or champions.

The research about how culture influences the implementation of Software Quality Management Systems was described. A number of hypotheses have been derived. Using Hofstede's four dimensions of national culture as an underlying discriminator, a revised questionnaire has been designed in order to determine the adoption of quality-oriented software processes and the awareness of them within the workforce. Fieldwork is currently in progress in Greece, Finland and UK.

References

1 Alavi M., Carlson P.: A review of MIS Research and Disciplinary Development, Journal of Management Information Systems, 8 (4) pp. 45-62

2 Mumford E., Hirschheim R., Fitzgerald G., Wood-Harper A.T (Ed.): Research Methods in Information Systems, Elsevier Science Publishers

3 Galliers Robert Ed.: Information Systems Research, Issues, Methods and Practical Guidelines, Blackwell Scientific Publications

4 Von Hellens L.A: Information systems quality versus software quality. A discussion from a managerial, an organisational and an engineering viewpoint. Information and Software Technology, 39, 1997, pp. 801-808

5 Ives B., Järvenpää S.: Applications of Global Information Technology: Key Issues for Management, MIS Quarterly, Vol 15 March 1991, pp. 33-49

6 Keen P.W.: Planning Globally: Practical Strategies for Information Technology in the Transnational Firm in Palvia S., Palvia R., Zigli R. (Eds) The Global Issues of Information Technology Management, Harrisburg Pennsylvania: Idea Group Publishing, 1992, pp. 575-607

7 Tractinsky Noam, Järvenpää: Information Systems Design Decision in a Global versus Domestic Context, MIS Quarterly/ December 1995, pp. 507-529

8 Hunter M. Gordon, Beck John E.: A cross-cultural comparison of "excellent' system analysts, Information Systems Journal, 1996, 6, pp. 261-281

9 Couger, J.D. Adelsberger H. (1988): Comparing motivation of programmers

and analysts in different socio/political environments: Austria compared to the United States, Computer Personnel, 11(4), pp. 13-17

10 Couger J.D., Borovitz I., Zviran M.: Comparison of motivating environments for programmer/analysts and programmers in the US, Israel and Singapore In Sprague, R/H. Jr (eds) Proceedings of the 22nd annual Hawaii International Conference on Systems Sciences, IEE Computer Society Press, Washington, DC, 1989, pp. 316-323

11 Ein-Dor P., Segev E., (1993): The effect of national culture on IS: implications for international information systems. Journal of Global Information Management, 1, pp. 33-44

12 Hofstede Geert: Cultures and Organisations, Intercultural co-operation and its importance for survival, Software of the mind, McGraw-Hill UK, 1994

13 Eriksson Inger, Törn Aimo: Introduction to IST Special Issue on Information System Quality, Information and Software Technology, 39, 1997, pp.797-799

14 Kendall Julie E., Avison David: Emancipatory research themes in information systems development: Human, organisational and social aspects in Human, Organisational. And Social Dimensions of Information Systems development (A-24) edited by Avison D., Kendall J.E., DeGross J.I., Elseviers Science Publishers, IFIP, North-Holland, 1993, pp. 1-11

15 Sanders Marty (Ed): The SPIRE Handbook, Better, Faster, Cheaper Software

Development in Small Organisations, The SPIRE project, European Community 1998

16 Curtis Bill, Heflleey William E, Miller Sally. Overview of the People

Capability Maturity Model, CMU/SEI_95-MM-01, 1995

17 Jack, Rickard. Personal Issues in SoftwareCost Estimation,

5th Software Quality Conference, 9-11 July, Dundee, 1997

18 Hodge B.J., Anthony W.P. Organisational Theory, Allyn and Bacon, Boston, 1988

19 French W.L., Bell C.H.: Organisation Development: Behavioural Science

Interventions for Organisation Improvement, Englewood Cliffs, NJ, Prentice-

Hall International, 1990

20 Johnson G., Scholes K.: Exploring corporate Strategy, 3rd edn, Hemel Hempstead, Prentice Hall, 1993

21 Goodman M.,: Creative Management, Hemel Hempstead, Prentice Hall , 1995

22 Senior Barbara: Organisational Change, Financial Times Management, Pitman

Publishing, 1997

23 Benjamin Robert: Managing Information Technology enabled Change

In Human, Organisational. And Social Dimensions of Information Systems development (A-24) edited by Avison D., Kendall J.E., DeGross J.I., Elseviers Science Publishers, IFIP, North-Holland, 1993 pp. 381-398

24 Cummings T.G., Worley C.G. (1993), Organisation Development and Change

(5th edn) St Paul, MN, West Publishing Company

25 Geletkanycz Marta A.: The salience of 'Culture's consequences': The

Effect of Cultural Values on Top Executive Commitment to the Status Quo

Strategic Management Journal, 1997, Vol.18:18, pp. 615-634

26 Orlikowski Wand J., Baroudi Jack J. Studying Information Technology in Organisations: Research Approaches and Assumptions, Information Systems Research 1991,2:1, 1-28

27 Land Frank ,The Information Systems Domain in Information Systems

Research, Issues, Methods and Practical Guidelines ed. Galliers Robert

28 Ralston David A., Holt David H., Terpstra Robert H., Kai-Cheng You

The Impact of National Culture and Economic Ideology on Managerial

Work Values: A Study of the United States, Russia, Japan and China

Journal of International Business Studies, 1st Quart. 1997, pp.177-205

29 Mohamed Walaa Eldeen and Siakas Kerstin V. (1995): Assessing Software Quality Management Maturity (SQMM): A new model incorporating technical as well as cultural factors, the 3rd International Conference on Software Quality Management SQM95, 3-5 April, 95, Seville, pp. 325-336

30 Siakas Kerstin V. The Effect of Cultural Factors on the Implementation of Software Quality Management Systems, 5th Software Quality Conference, Dundee, 9-10 July, -96

31 Newman Karen I., Nollen Stanley D: Culture and congruence: The Fit between

Management Practices and National Culture, Journal of International Business

Studies, 4th Quarter 1996, pp.753-777

IPSSI: A European Methodology on PSP

Yingxu Wang, Howard Duncan*, Minna Kaartinen, Hasse Sjöström and Paula Kökeritz
Centre for Software Engineering
Dept. of IT Systems, IVF
Argongatan 30, SE-431 53 Mölndal, Sweden
Tel: (031) 706 6174, Fax: (031) 27 61 30
E-amil: {ywg, mka, hs, pk}@ivf.se

* School of Computer Applications
Dublin City University, Dublin 9, Ireland
E-mail: Howard@compapp.dcu.ie

Abstract

It is a recent trend in adapting existing software process models at the personal process level. The Personal Software Process (PSP^{SM}) is a typical personal process model derived from CMM^{SM} and related work. A European project on Improving Professional Software Skills in Industry (IPSSI) has been founded. The project is aimed at developing a European process model for individual software engineers in small and medium sized enterprises (SMEs).

This paper describes the architecture of the IPSSI project and its intermediate outcomes. Requirements of European software industry for the IPSSI personal process framework are analysed. The IPSSI process framework, data capture methodology, measurement attributes, and an IPSSI support tool are explored. A mapping between the frameworks of IPSSI and PSP is provided.

Key Words: Software engineering, software process improvement, personal process, PSP, IPSSI

Introduction

Software process technologies can be classified into three levels: the organisational [1,2], project's [3], and personal [4-6] processes. The personal process of software development is a new technology that encourages individual software engineers to adopt the best practices in software engineering. The personal software process (PSP^{SM}) [4-6]

is the first and typical approach to disciplined personal software process modelling. In his book, Watts Humphrey said: "The PSP's sole purpose is to help you be a better software engineer [4]."

The European Commission has founded a project IPSSI (Improving Professional Software Skills in Industry) [7-11] within the ESSI programme in the framework of ESPRIT. The project aims to provide a process improvement framework for use of tailored and adapted PSP technology by individual software engineers in small and medium sized enterprises (SMEs).

Data gathered from software engineers who have practised the PSP method show the following benefits:

- Improved size and time estimating accuracy
- Lower defect densities
- Dramatic reduction of defects during compiling and testing
- Improved productivity

However, software process improvement in Europe has been focused on organisational level. There is an absence of support for process improvement at the individual level. As a result, the following problems have been identified in the European software industry:

- There is a gap for providing individual’s process framework and technology in software process improvement;
- The SMEs need a suitable method to improve their software engineers capability for developing software with higher quality and productivity;
- The SMEs need a set of personal process training materials that is suitable to the European software industry;
- The SMEs need a computer-aided tool to support the personal process training with efficiency.

The focus of the IPSSI project is on improving individual software engineers’ skills thus enabling bottom-up process improvement. The IPSSI project aims at:

- Provide a process improvement framework for use by individual software engineers;
- Develop a set of materials which can be used to train software developers to use personal processes;
- Create a body of trainers who are capable of couching the widespread applications of this disciplines;

- Develop a tool to enable developers to record data of their performance while using IPSSI methodologies;
- Carry out case studies to gain experience in personal process improvement;
- Learn how software development activities can be improved by using the personal processes;
- Learn how the personal processes can be used to improve the management of engineers and projects.

In the following sections, the architecture of the IPSSI process model and the IPSSI support tool will be described. Industry requirements for the IPSSI framework are reported.

2. Architecture of IPSSI

This section describes the structure of the IPSSI personal process framework. The IPSSI data capture methodology, measurement attributes, and support tool are explored. A mapping between the frameworks of IPSSI and PSP is provided.

2.1 IPSSI Project Structure

IPSSI is aimed to develop a personal process model that is suitable for the European software industry, based on the experience gained and lessons learned in applying the PSP. An overall structure of IPSSI is shown in Figure 1.

The kernel of IPSSI is its process model, and data capture and measurement methodology. Based on the IPSSI personal process model, a set of training materials is developed for assistant software engineers, programmers and project managers to apply the IPSSI processes and methodologies. An IPSSI software tool has been developed to support the training and application of IPSSI methodology in the software industry.

2.2 IPSSI Process and Data Capture Methodology

A structure of the IPSSI process model is shown in Table 1. IPSSI has defined three process levels, with level 0 as the baseline. IPSSI focuses on the planing and quality management processes. A set of measurement is designed for showing what personal software process data will be captured and analysed.

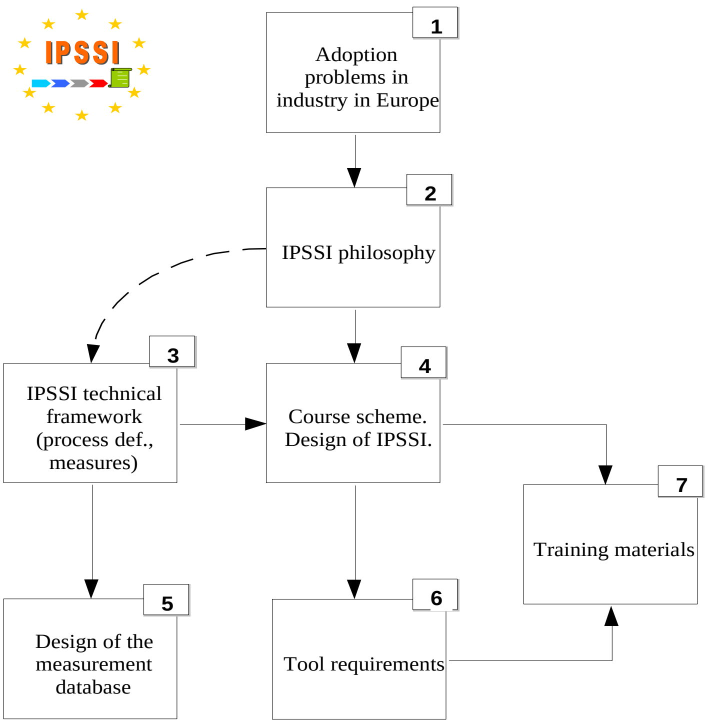

IPSSI
1
Adoption problems in industry in Europe
2
IPSSI philosophy
3
IPSSI technical framework (process def., measures)
4
Course scheme. Design of IPSSI.
7
Training materials
5
Design of the measurement database
6
Tool requirements

Figure 1. The structure of IPSSI framework

Table 1. IPSSI processes and data capture requirements

Level	Process	Measurement
0 (Introduction)	Baseline	
		- Current processes - Time records - Defect/KLOC before compile - Defect/KLOC in unit test - Defect/KLOC in system test
		- Coding style conformance
1 (Basic)	Planning	
		- Estimated size - Real size
		- Effort estimation - Size estimation - Schedule estimation
2 (Advanced)	Quality management	
		- Defects found in review
		- Design errors found

		- Productivity - Design quality

A mapping between the personal process models of IPSSI and PSP is shown in Table 2. Table 2 indicates that IPSSI is a tailored PSP for adapting to the needs of European software industry. IPSSI puts emphases on three processes, such as planing, estimating and quality management; and three measurable attributes, such as size, effort, and defects. This approach allows SMEs and software engineers focus on important personal processes with well-defined measurement.

Table 2. Mapping between the IPSSI and PSP process models

Level	IPSSI Processes	PSP Processes
0	Baseline	Baseline
	- Description of current processes	PSP0 – Current process
	- Time records - Defect records - Coding style	PSP0.1- Process improvement
1	Planning, scheduling and estimating	Planning
	Size: estimated/real	PSP1 – Size estimating and test report
	Estimation: - size - effort - schedule	PSP1.1 – Task and schedule planning
2	Quality management	Quality management
	- Design review	PSP2 – Code and design review
	- Design quality	PSP2.1 – Design templates
	- Productivity	PSP3 – Cyclic development

2.3 IPSSI Measurement Attributes

The main measurement attributes adopted in IPSSI are size, effort, and defects of software projects.

- *Size*. Line of code (LOC) of a developed product.
- *Effort*. The time a software engineer spends in a personal project.
- *Defects*. The errors and bugs appeared in the developed product, and during development phases.

The main measurement attributes and their relationships with the IPSSI processes are shown in Figure 2.

2.4 Design of an IPSSI Support Tool

The IPSSI support tool is designed to support the following functions:

- IPSSI process introduction
  - Processes
  - Explanations

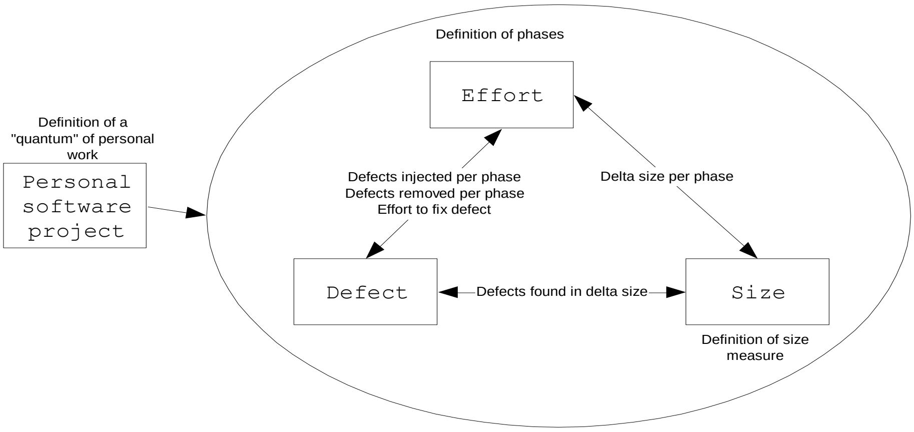

Definition of phases
Effort
Definition of a "quantum" of personal work
Personal software project
Defects injected per phase
Defects removed per phase
Effort to fix defect
Delta size per phase
Defect
Defects found in delta size
Size
Definition of size measure

Figure 2. IPSSI processes and measurement attributes

- Data capture for each process

- Data analysis for each process
  - Current vs. average
  - Current vs. history
  - Strengths
  - Weaknesses

- Help (to show how to use the tool)

A structure of the IPSSI tool is shown in Figure 3. The main idea is to keep data gathering and data storage/analysis separate. For doing so, the IPSSI tool suite consists of a data gathering tool, a data submission tool, and a data analysis tool. In such a configuration, a light on-line data gathering tool can be implemented and activated during individual's programming process, and the privacy of personal data can be guaranteed.

A number of existing PSP tools have been analysed and evaluated [11-12]. The functions of the IPSSI tool are described below:

Data gathering. ***The IPSSI data gathering tool supports the following operations:***

- Easy to use on the desktop;

- Allow users to work in various personal processes;
- Measure all data necessary to be derived, such as:

the effort per process phase;

the size of the developed product;

the number of defects found per phase;

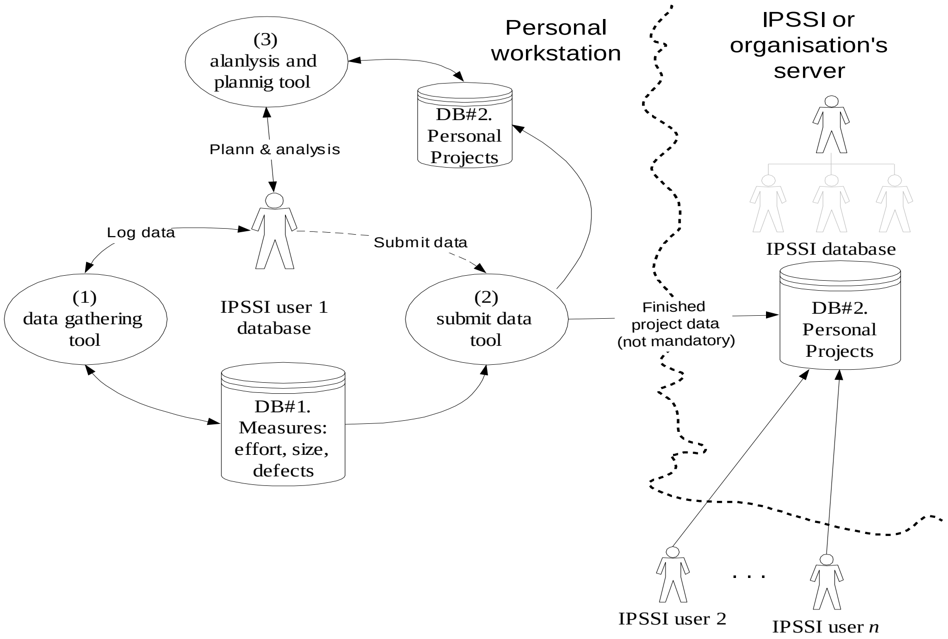


Figure 3. The structure of the IPSSI tool

- Allow data logging with as few user interactions as possible;
- Accept events from the operating system. For example a compilation event, or the defect found data.

Data submission. ***The IPSSI data submission tool supports the following operations:***

- Isolate properly the two database structures, making each one as much independent of the other as possible;
- Allow users decide if personal data will be sent to the public IPSSI database;
- Automatically update the personal project database on the local workstation.

Data analysis. ***The IPSSI data analysis tool supports both data analysis and project planing functions. For data analysis, the IPSSI tool enables the personal process performance be analysed and reported, including plotting the following parameters:***

- Effort per phase
- Defects injected per phase
- Defects removed per phase
- Total size
- Planning accuracy
- Productivity
- Phase yield
- Defect densities
- Defect removal efficiency per phase

For the planning functions, the tool supports store, compare and reporting estimated and real data in project planing. Facilities for plan and schedule tracking are also provided.

3. Analysis of IPSSI Regional User Needs

A questionnaire of the European IPSSI project on PSP has been developed [8], and a survey of regional user needs has been conducted in Sweden [10]. The objectives of this survey are to identify the needs on PSP in Swedish software industry, and to develop a personal process methodology that is suitable to the industry.

This section summarises the regional requirements towards IPSSI methodology and training methods based on the IVF surveys.

3.1 Awareness and Interest in Personal Process

The PSP is well aware in Swedish software industry. Some training organisations have been running PSP courses for a certain time. A number of universities have established PSP causes in their computing or information systems curricula.

According to the distribution of profession among the software engineering occupations as below:

- Software engineer and/or programmer
- Testing engineer
- Quality assurance engineer
- Project manager
- System analyst
- Chief executive

The largest group, who has expressed interest in IPSSI and PSP methods, is the project managers, followed by software engineers. This data indicates that personal process is usually required by the professional they would directly use it in everyday work.

3.2 Organisations' Priority in Process Improvement

For answering "What is your organisation's priority in software process improvement - at organisational, project's or individual's process levels?" The feedback was mainly at project level, but was less emphasised on individual level. This would be a negative factor on promotion of the personal processes; while on the other hand, it can be seen as an opportunity for an IPSSI personal process model that focuses on personal processes.

3.3 Popular PSP Processes

Concerning the usefulness and effectiveness of PSP processes, the top five PSP processes that were thought most useful in practice were:

- Design review
- Schedule estimation
- Process improvement proposal
- Personal planning in software development
- Program size estimation

3.4 Requirements for Adding Features to PSP

On commenting what the industry need to extending PSP processes, a number of potential areas have been identified as follows, by listing in descending importance as shown in the survey:

- Requirement analysis process;
- Reuse process;
- A defined capability scale for show programmers' current levels and future improvement;
- Tailorability of generic IPSSI training materials;
- An IPSSI tool enable self-learning;
- Tool functions for reporting metrics as well as programmers' capability levels and improvement history.

3.5 Expectations on IPSSI

On questioning "what are your expectations on an IPSSI training", answers weighted in a descending order are:

- Design template
- Coding standard
- Design review
- Defect type standards
- Time recording
- Program size measurement
- Defect recording
- Personal planning
- Code review
- Process improvement proposal
- Program size estimation
- Test reporting
- Schedule estimation

According to the survey it is also found that the team processes [3] are largely required by software project managers.

3.6 Open Issues on IPSSI/PSP

Along with the very positive support in the survey for the IPSSI project, there are a number of open issues found from the software industry. The main open issues are summarised below as important research topics in developing personal process models.

- In a software development organisation, when a software engineer is only responsible for limited roles and specific processes on a project, the one-dimensional PSP-like model would not be suitable. Therefore we need a two-dimensional IPSSI model that consists of a process dimension and a capability dimension. By using the 2-D model, each process can be assigned to any of the software engineers in a project team, and each process has to be evaluated against another dimension of capability levels. This is a more generic and flexible model framework for personal processes;

- *Adoption of the personal process is very much dependent on project level processes and project leaders' agreement. The organisational and cultural changes would be one of the barrels for implementing the personal processes in the industry;*

- The current personal process models, such as PSP and IPSSI, have been focused on measurement. An argument was that whether only enhanced measurement of the personal processes without any improvement and adaptation can solve software engineering problems at the individual level?

- How do we interface personal processes with team and organisation processes? How do we solve possible problems of conflict and keep consistency between IPSSI/PSP and the organisational process models such as ISO/IEC TR 15504 and CMM?

5. Conclusions

The main deliverables of the IPSSI project are:

- An IPSSI personal process framework, which is suitable for the European software industry;
- A set of materials which can be used to train software engineers for the personal software process disciplines;
- A computer-aided IPSSI tool for supporting applications and training;
- A series of training seminars, on site training and case studies, by these a repository of the best personal processes for software engineering will be established.

This paper has presented the architecture of IPSSI, and its orientation to the European software industry. Regional software industry requirements have been surveyed and analysed. Based on these, the IPSSI personal process model, its measurement, and a supporting tool are developed. In addition to the progresses of IPSSI in personal process modelling, a number of open issues has been discussed and is under investigation.

Acknowledgements

This project is founded by European Commission through the ESSI programme No. 27453. The authors would like to thank colleagues Gerry Coleman, Rory O'Connor, Stefan Schuster, Maurizio Morisio, David Escala, and Géraldine Pichon for their work. We would like to acknowledge the following partner organisations for their invaluable support:

- Dublin City University, Ireland (project co-ordinator);
- IVF - The Swedish Institute for Production Engineering Research;
- ESI - The European Software Institute, Spain;
- Université Thomson, France;
- Politecnico di Torino, Italy.

References

[1] Paulk, M.C., Curtis, B. and Chrissis, B. (1993), Capability Maturity Model, Version 1.1, *IEEE Software*, July, pp.18-27.

[2] ISO/IEC TR 15504-2 (1998), Information Technology – Software Process Assessment - Part 2: A Reference Model for Processes and Process Capability, ISO/IEC ISO, Geneve, pp. 1 - 39.

[3] Humphrey, W.S. (1999), The Team Software Process, Proceedings of International Conference on Production Focused Software Process Improvement, VTT, Finland, pp.11-12.

[4] Humphrey, W. S. (1995), A Discipline for Software Engineering, SEI Series in Software Engineering, Addison-Wesley Publishing Company, Inc., USA.

[5] Humphrey, W.S. (1997), Introduction to the Personal Software Process, Addison Wesley.

[6] Humphrey, W.S. (1996), Using a Defined and Measured Personal Software Process, IEEE Software, May, pp.77-88.

[7] IPSSI project (1999), Introduction to IPSSI -- One-Day Programme, IPSSI Technical Report, ESPRIT Project No. 27453, pp.1-5.

[8] IPSSI Project (1999), IPSSI Questionnaire V.2.0 - Improving Professional Software Skills in Industry, ESPRIT Project No. 27453, pp.1-14.

[9] Wang, Y. (1999), IPSSI Support Tool Requirement Definitions, IPSSI Project Technical Report D.2.2.2, IVF, pp. 1 - 3.

[10] IVF (1999), Survey on Regional User Needs, IPSSI Project Technical Report, D.3.1, pp. 1 - 3.

[11] Geraldine Pichon (1999), Existing PSP Tools Evaluation, IPSSI Project Technical Report, ESPRIT Project No. 27453, pp. 1 - 6.

[12] East Tennessee State University (1997), Personal Software Process Studio User's Manual: Computer-Aided Software Engineering for the Personal Software Process, pp. 1-37.

[13] O'Beirne, P. &Sanders, J. (1997), Does the PSP Deliver Its Promise? Proc. Inspire II, SGES Publications, Gothenburg, 1997.

Session 13
SPI and Object Orientation

Chairman
Pekka Forselius

STTF, Finland

Improvement of the Quality in the Software Development Process by the Introduction of UML

Arthur Görges, Minsheng Liu
Hüngsberg AG,
Linienthalsraße 2
D-85399 Hallbergmoos, Germany
E-mail: {apg, mil}@daxware.de

ABSTRACT: *This paper presents the result of the experience acquired from the ESSI project called OFTPIVE.PIE. In this project the object-oriented methods with UML and a new paradigm have been introduced to the application environment of the software development process and to improving the software development process. By executing the project, the weakness in the process of our software development have been recognized, so that the process of software development has been redefined and improved. The use of the forward, reverse and round trip engineering can reduce the development time of the software product. It is shown that the introduction of UML, CASE tool and a new paradigm is beneficial to the software development process.*

1. Introduction

Huengsberg AG is an international company in the field of information and communication technology in automotive industry, based on the Organisation for Data Exchange for Tele-Transmission in Europe (ODETTE). In the past the company has developed, manufactured and marketed a wide range of hardware and software products in the ISDN for automotive industry. We developed and provided a variety of software products for our customers, and emphasised the importance of servicing for the customers and meeting customer's requirement. However we did not pay much attention to developing the methodology of our software product development. With increasing complexity of our products resulting from customer's requirements, we found that a permanent improvement of the software development process is urgent, if we want to provide our customers with better service and high quality products.

The goals of this project, which was funded by the European Systems and Software Initiative(ESSI), are to improve the software development process and to reduce the errors of analysis, design and implementation in software development by using object-oriented methods with UML in our application environment. So we are able to make our customers more satisfied with high quality and with the best cost-to-benefit ratio. In this paper the project is firstly described in brief, then we report the results and experience obtained by executing the PIE project .

2. Description of the Project

Baseline Project:

In the European automotive industry file transfer based on ODETTE File Transfer protocol uses simple point to point links, today. There is a number of file exchange systems between different manufactures and suppliers. As the CAD transfer volumes is increasing, these systems can not match the customer's needs. On the other hand the use of these systems is also expensive, e.g. a company has to pay in full for 24 hours, even if the company uses only the system a few hours in a day. The application of electronic communication for file transfer is necessary and important for the automotive industry.

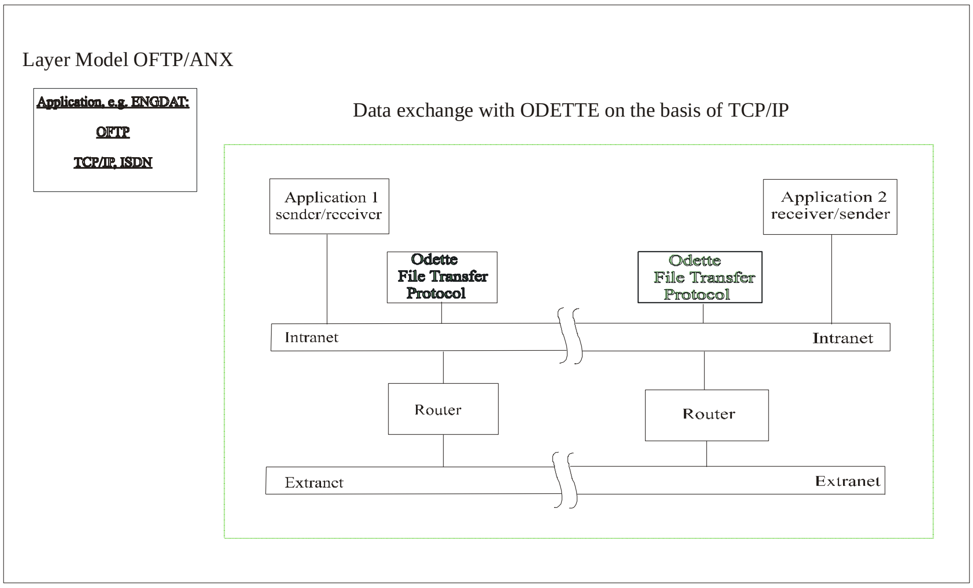

Layer Model OFTP/ANX
Application, e.g. ENGDAT:
OFTP
TCP/IP, ISDN
Data exchange with ODETTE on the basis of TCP/IP
Application 1 sender/receiver
Application 2 receiver/sender
Odette File Transfer Protocol
Odette File Transfer Protocol
Intranet
Intranet
Router
Router
Extranet
Extranet

Fig.1: File transfer through TCP/IP in the baseline project.

On the basis of the international networks and Corporate Networks the baseline project [3] has been developed for the file transfer in the automotive industry. The ODETTE file Transfer Protocol takes an interface function for a standardized and efficient communication over the network as shown in Fig. 1.

The software product of the baseline project is based on logical connections between product providers and manufactures, so the software product provides our customers with much efficient service. In the experiment two server modules were designed in the conventional way using Microsoft C++ and Micro FoxPro as programming language. One transmission protocol is subject to the test-wise re-implementation by making use of object-oriented modelling and programming.

Definition of the PIE Project:

The PIE project is proposed to evaluate a new software development method, namely, object-oriented method by using UML (Unified Modelling Language) [5] and CASE

Tool for software development process in the automotive industry. The programming language JAVA is chosen for the support environment [4].

In order to evaluate the object-oriented method with UML, we applied it to re-implement modules of the baseline project. tool. The experiment consists of the following components: Definition of experiment, Sending and receiving process, Logging activities, Interprocess communication and Investigation of data security as shown in Fig. 2.

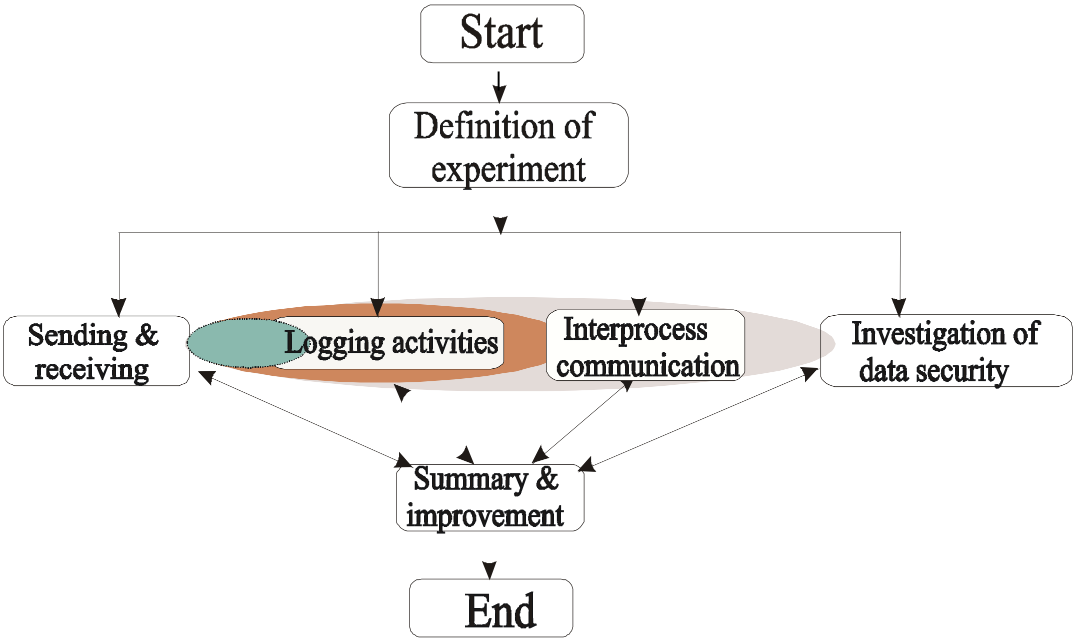


Fig.2: Composition of the PIE experiment: Definition of experiment, Sending- and receiving process, Logging activities, Interprocess communication and Investigation of data security

3. Implementation of the Improvement Actions

In order to achieve the goals of the project the following main activities have been taken for implementing the experiment:

- **Analysis and definition of the software development process:**

  In the project program a process of software development, which consists of four phases, was planned for the PIE project. Afterwards we realized that the process is not fit to the environment of our software development. With the aid of the consulting company 3 SOFT the current application environment of development process is firstly analysed by using the new method kit [1, 2]. Some weak areas in the software development process have been discovered.

Combining the existing environment with object-oriented development methods, the new process of the software development has been defined carefully again and completed in detail.

The Fig. 3 shows the improved process of the software development as an example. The improved process consists of 8 phases such as Analysis, design, implementation, Integration test and so on. By using the improved process the software development can be easily implemented by the iterative and incremental method.

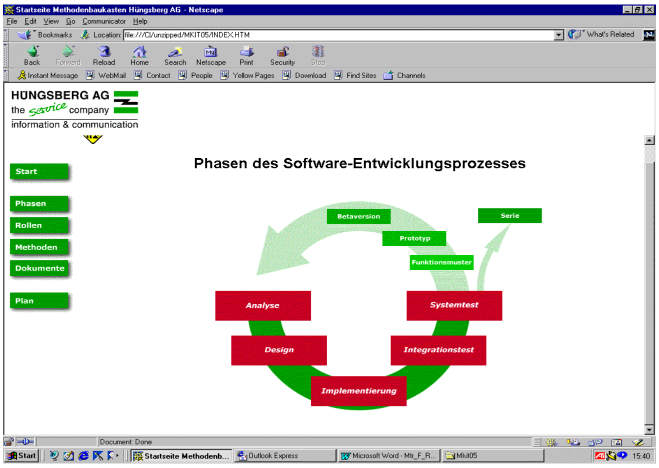

Startseite Methodenbaukasten Hüngsberg AG - Netscape
File Edit View Go Communicator Help
Bookmarks Location: file:///C|/unzipped/MKIT05/INDEX.HTM
What's Related
Back Forward Reload Home Search Netscape Print Security Stop
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
HÜNGSBERG AG
the service company
information & communication
Start
Phasen
Rollen
Methoden
Dokumente
Plan
Phasen des Software-Entwicklungsprozesses
Betaversion
Serie
Prototyp
Funktionsmuster
Analyse
Systemtest
Design
Integrationstest
Implementierung
Document: Done
Start
Startseite Methodenb...
Outlook Express
Microsoft Word - Mtr_F_R...
Mkit05
15:40

Fig. 3: The improved process of the software development.

Roles in the software development process are defined as a set of related tasks of one person, each roles has the responsibility to achieve the predefined objectives. Moreover the documents and its plan of the project, the methods used in the project have been defined.

- **Selection of a CASE Tool:**

There are different CASE tools existing in European market. For selecting a suitable CASE tool, we reviewed firstly them, then after considering the requirements of the PIE project, two CASE tools: Rational Rose and Together, were chosen for further test by using the modules of the PIE project.

When checking the CASE Tool the following factors are considered:

User-Interface: A CASE tool is designed to assist in software development, and it should be logical and easy to use by designers without long training periods. A tool should support familiar GUI elements and UML, and integrate other applications such as Microsoft word, etc.

Workflow: Because of the weakness in the process of the analysis and the design phase, we regarded workflow as an important factor. A CASE Tool is able to model workflow of a new system in the field of the new system and to represent the software development process.

Code generation: On the basis of logical diagram a CASE tool must generate proper code in an appropriate format which implements the product model.

Reverse Engineering: Reverse engineering is the process of creating a model by analysing source code. As a software supplier, we offer our customers various products. It is often that we are asked to satisfy some special requirements by customers. A CASE tool has to allow to modify easily from analysis to design and to implement, and back to analysis again. The iterative style of development allows designers to begin with a set of known requirements, then evolve as project parameters change or new requirements are added and the project modelling is modified. The reverse Engineering and forward engineering are very important features for the dynamic development process, so that we alter the implement, assess the changes and incorporate them in the design.

The result of the studies indicate that the general functionality of these tools is similar. The CASE tool Rational Rose has more functions than the CASE tool Together. So Rational Rose is chosen as the suitable tool for the demands of the PIE project and is the choice for object-oriented software development in our company. Such a CASE tool provides us significant benefits in the speed of our developing new applications and also facilitates maintenance of the application throughout their life cycle.

- **Training Activities:**

  Software developers took part in the seminars about the object-oriented software development, unified modelling language, design patterns and OO-programming and software metrics. They gained a lot of knowledge in the object-oriented analysis, design, implementing and metrics.

- **Experiment:**

  The UML consists of different diagrams [5]. In this project the Use Case diagram, Class and package diagrams, Sequence diagrams, Activity diagrams and Component diagrams have been used for managing the project and for the process of the analysis, design and implementation. The following examples illustrate the application of the UML for the analysis, design and implementation.

  Documenting the behaviour and requirements with UML: The first step in the development of a software products is to achieve a understanding of the problems and define the behaviour of the products. This begins with the assessment and documentation of the product requirements. In this project the use case model is used in documenting the behaviour and requirements of the project. The use case model illustrates the system's intended functions (use case), its surroundings (actors) and relationships between the use case and actors (use case diagrams). It provides a view of the system structure and one starting point for design. Fig. 4

shows the actors in the project and the use case diagram of the PIE project. It provides a detailed view of the project for the communication between the development team members and customers.

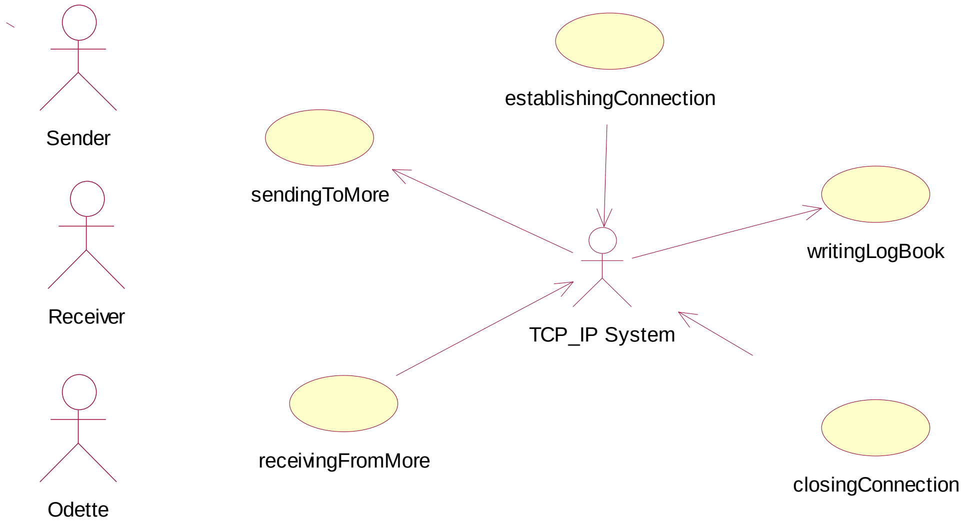


Fig. 4: The actors in the project and the use case diagram of the sender and receiver process, Logging activities and interprocess communication. This presents a view of the functions of the TCP/IP subsystem.

Software design with UML: A good software design must support the below abilities: Comprehensibility, maintainability and extensibility. In this project we employed the class and package diagram, sequence diagram, activity diagram and component diagram for the software design. The Fig.5 and Fig.6 as instances display the class diagram and the sequence diagram.

The class diagram of three components in the project is shown in Fig. 5. The class diagram provides a view of the classes in the logical view of the design , which illustrate how the classes relate. The class diagrams are also the foundation for code generation.

Class diagram and package diagrams are static. They are not adequate to determine whether the design is adequate to meet the requirements. From the class diagrams we cannot learn about the behaviours of the system. Sequence diagram can meet this need. The elements of sequence diagrams are objects and messages. The Fig. 6 shows the sequence diagram of the project DAX 2000 as example, where the boxes with underlying dashed line indicate objects.

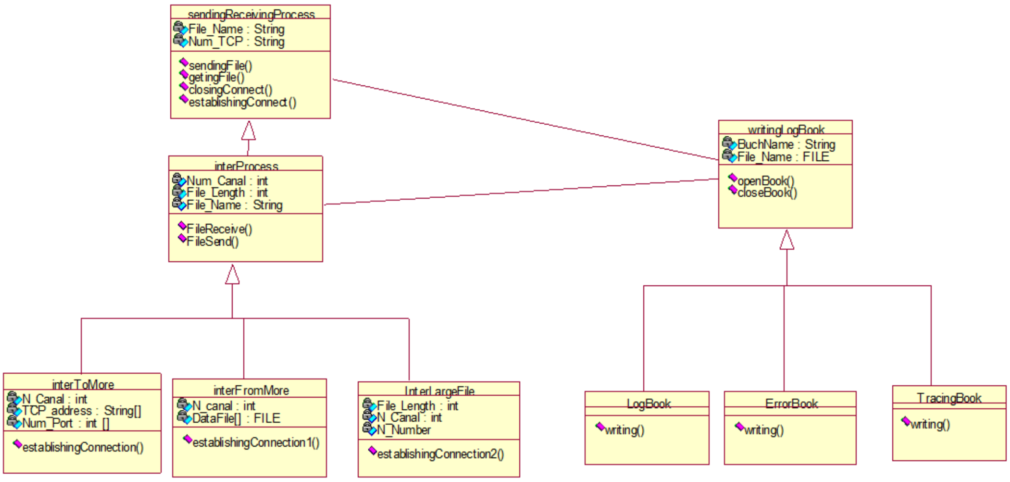


Fig. 5: Class diagram of the project components: sender and receiver process, logging activities and interprocess communication. It presents the view of the classes in the model and the interactions among them.

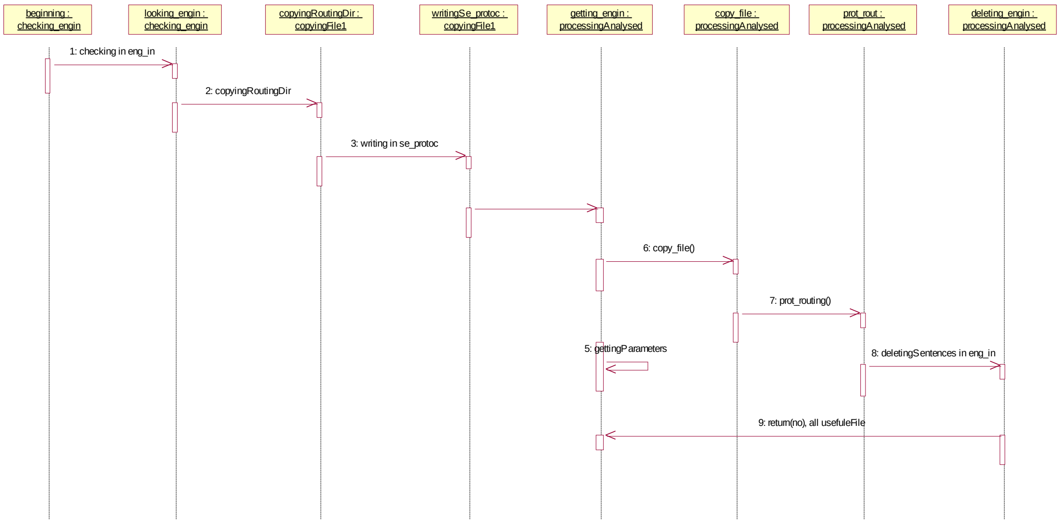

beginning : checking_engin
looking_engin : checking_engin
copyingRoutingDir : copyingFile1
writingSe_protoc : copyingFile1
getting_engin : processingAnalysed
copy_file : processingAnalysed
prot_rout : processingAnalysed
deleting_engin : processingAnalysed
1: checking in eng_in
2: copyingRoutingDir
3: writing in se_protoc
6: copy_file()
7: prot_routing()
5: gettingParameters
8: deletingSentences in eng_in
9: return(no), all usefuleFile

Fig. 6: Sequence diagram of process of the incoming file in DaxENGdat of the project DAX 2000. It shows the object interactions arranged in time sequence.

Sequence diagrams are derived from the development of use cases; they present the objects, messages and object interactions arranged in time sequence.

In the development process of each component the above phases have been used repeatedly. The iterative and incremental method has been easily realised by using the round trip engineering. The Fig. 4 shows the user interface of the sender.

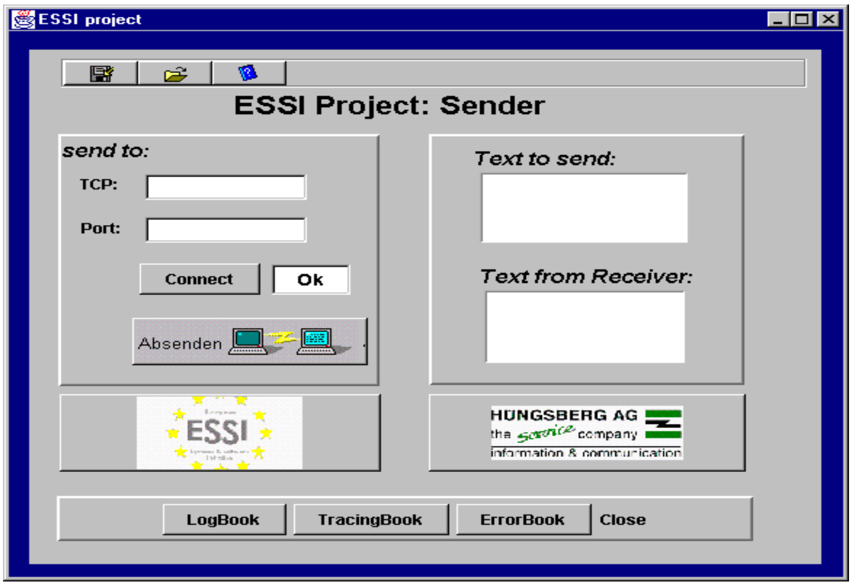


Fig. 7: The user interface of the sender.

4. Result and Conc1usion

By executing the PIE project we have introduced the new method to the process of our software development. The process of the software development has been improved and complemented. A standardised software development process has been established. The software developers have realised that how-know becomes very quickly out-of date in the IT sector. They should keep on learning in lifelong and now are more motived for new technology.

The Fig.8 illustrates the application of the UML and the CASE Tool for the development phases. It shows how the requirements of customer's are realized through the phases: analysis, design, implementation and code test by using the CASE tool. The impact of the UML on the each phase of the software development process is represented by the colour. The darker colour presents the greater impact of UML on the phase. It indicates that the UML is suitable to the analysis and design phase. It has also influence on implementation and code test phase.

By the execution of the experiment, we gathered the following information:

- The UML is a very useful language for modelling a software development process and enables developers easy to communicate with users of products and members in a development team. With the aid of UML and proper CASE tool, the software development process can flexibly and quickly be modified according to the new requirements of customers.

- By using the UML each step in the software development process can be represented in detail, so the quality of the development process can be easily examined and controlled.

- The object-oriented thinking is a foundation for a developer to use a UML properly and smoothly.

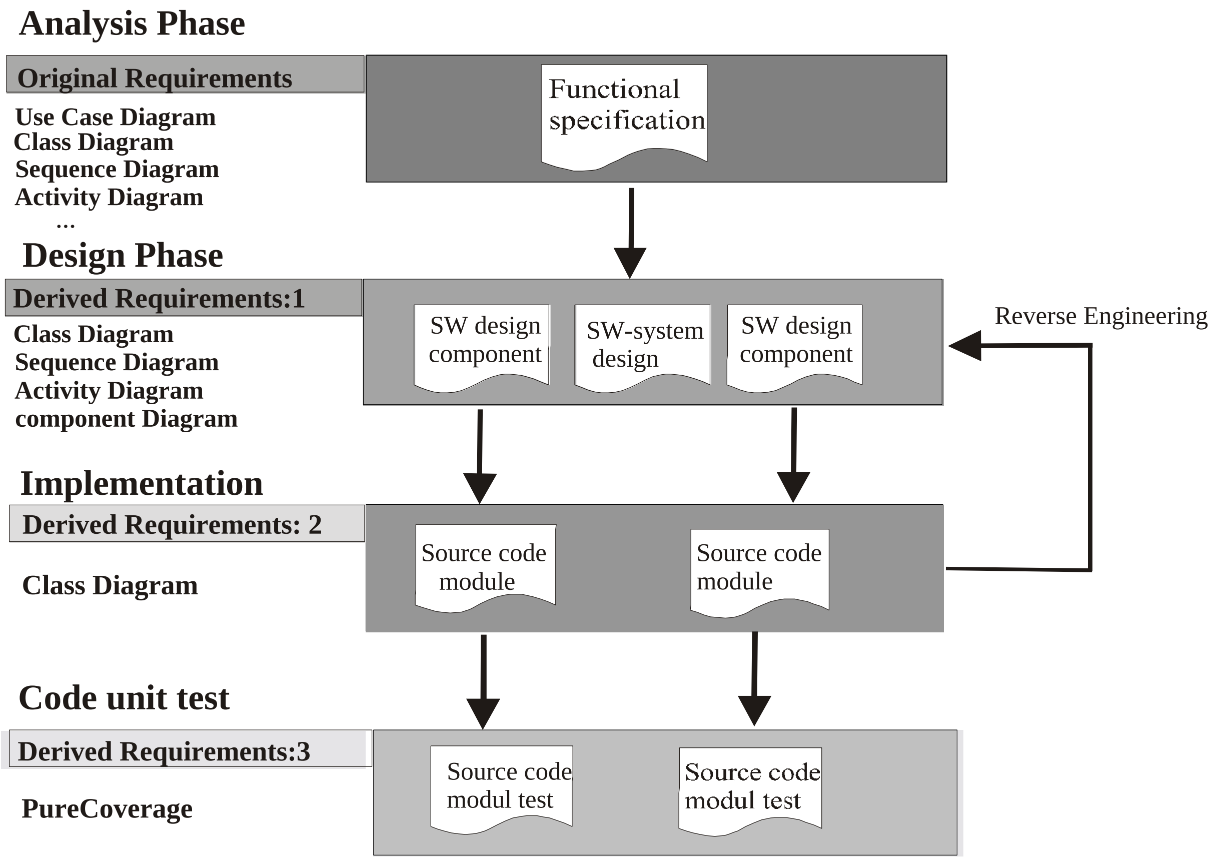


Fig. 8: The impact degree of the UML on the software development process: the darker colour presents the greater impact.

- After introducing the UML and CASE tool, the code of class level can be generated by class diagrams, the development period of modules in the project can be reduced
- The CASE Tool „Rational Rose“ is especially suitable for modelling software development process. The generation of source code works properly and the reverse engineering has good functions. The iteration and increment development process can be more easily implemented by using the round-trip engineering
- Combining UML with the new paradigm used to manage the software development, the software development process can be analysed and administered in a proper and efficient manner. Especially the weak areas of the software process in our company have been recognised, this is a basis for further

improvement

- The generated code consists of the class definitions. It is not the complete executable code. This must be implemented by hand, and so it is necessary to generate clear and concise code type for further programming.
- All of the diagrams in the UML presents the same model in varied views on the basis of different detailed aspects. The connection and dependence among diagrams have been neglected.

Conclusion:

In this report we present the result of the OFTPIVE.PIE project. It is indicated that the introduction of UML and CASE Tool results in positive impacts on software development process. Especially the UML is suitable to apply for modelling the software development process and for documenting the software analysis and software design. The development period of the product can be reduced by the forward, reverse and round trip engineering. This method has been used in modelling the project *DAX 2000* in the process of our software development. We will continue applying this method for the process of other software product developments.

5. References

1. H. Balzert: ”Lehrbuch der Software-Technik”, Spektrum Akademischer Verlag Heidelberg, Berlin Berlin, 1988.
2. A. Barthel, B. Hindel:“*The Method Kit: A new Paradigm for Process Definition*”, CONQUEST’ 98, pp.192-200, Sept. 1998 in Nuernberg, Germany.
3. W. Huengsberg: ”*Ein Extranet in Internet für die Unternehmenskommunikationn in der Automobilindustrie*”, Sonderdruck des Beuth Verlags aus edi-change, No. 2 1997.
4. R. Gema: ”*Gruppenbild, Java-Entwicklungsumgebungen im Praxisvergleich*”, Magazin für computer technik, pp. 180-189, No. 5 1999.
5. B. Oestereich: “*Objekt-orientierte Softwareentwicklung*”, R. Oldenbourg Verlag münchen Wien, 4. Auflage, 1998.
6. A. Goerges, M. Liu: “Tool Selection Review Report”, the report of the ESSI project, July 1999.

Authors:

Arthur Görges is Chief Operating Officer Research & Development at Huengsberg AG. Since 1983 he has been working on computer technology, networking and electronic communication. His main areas of interest: project management of software development and the application of ISDN, internet technology for electronic communication in the automotive industry.

Minsheng Liu received the PhD in applied Physics at the University of Frankfurt am Main in 1998. He is working as a software developer at Hungsberg AG. His main interests are: object-oriented programming with JAVA; UML, CASE Tool, TCP/IP and the application for software development process.

Improvement of Extendibility and Modifiability of Embedded Software Through Application of Object-Oriented Design EMESO

Dr. Manfred Dresselhaus
Reis Robotics, Obernburg

Abstract

REIS ROBOTICS is forced to permanently improve its products in order to maintain its competitiveness. Continuous introduction of innovative solutions as well as fulfillment of customer reeds require, that the robot control system can be efficiently extended and modified. To achieve this, REIS has started the EMESO project to improve its software engineering (SE) process. The main objective of this project is the introduction of a new SE methodology in the software development process of the robot control software. Object oriented (OO) methods are promising approaches to achieve these requirements. The experiment is carried out to ensure that an OO approach is appropriate. This paper describes the structure of the EMESO project and

the main activities carried out. The presentation of the results mainly focuses on the lessons learnt. These didn't only concern technical aspects such as improvements in the code structure and the introduction of new SE tools but also organizational and people related aspects.

Introduction

The EMESO project is carried out as a process improvement experiment (PIE) and is funded by the European Commission in the frame of the European Systems and Software Initiative (ESSI).

Such a PIE is principally built up as shown in **fig. Dr.1**. The experiment is carried out according to a so called baseline project that in the actual case is the development of a new extended version of the existing REIS robot control. The baseline project itself is not part of the PIE. The PIE only covers a relative small part of the development process in which the experimentation is carried out. Before starting the PIE, a detailed analysis of the starting scenario, i. e. the situation before starting the experiment, is carried out. During the experimentation phase a permanent information exchange between the baseline project and the PIE happens. After finishing the experimentation phase a detailed analysis of the resulting scenario, i. e. the situation after finishing the experiment, is performed. The results are after that disseminated in order to make them available to a wider public.

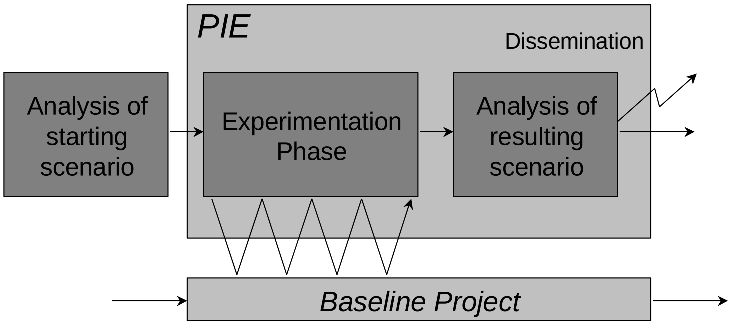


Fig. Dr.1. *Basic structure of a process improvement experiment (PIE)*

The introduction of an OO approach concerns all sectors of the software development process and comprises technological, organizational, human related aspects. In the scope of EMESO it is not possible to address all of these aspects. Based on the current status of SE practice at REIS, the most critical sector for improving the extendibility and modifiability of software is design and implementation.

EMESO tests OO design and implementation of software in the scope of the baseline project. Since the baseline project is of high importance for REIS, a parallel development of certain software modules has to be made in the scope of EMESO. Therefore, the following approach is carried out:

The OO methods and tools applicable for real-time embedded software are analyzed in order to identify the most appropriate ones.

In the scope of EMESO an experimental system with a small OO control kernel is built. This kernel is defined in order to identify a minimum amount of software which has to be re-developed. This kernel has to include some critical and also some typical modules with reference to computation time, memory resources and use of system interfaces.

The efforts needed for the extension in the experimental control are compared to the efforts needed for the extensions in the new control software.

Since the quality of the software for the control system is one of the main competitive aspects for REIS, careful testing of the implemented experimental system must be done.

The project EMESO is one single step in the planned long-term improvement process. However, this step is the most critical for successful carrying out of the required corrective action plan, since it has to provide clear answers on effectiveness of the OO methodology in the REIS SE practice.

The structure of this paper follows the structure of the PIE. After a short presentation of REIS ROBOTICS and its business and products in **chapt. 2**, in **chapt. 3** follows a description of the analysis of the starting scenario at the beginning of the experimentation phase. **Chapt. 4** contains the implementation and improvement actions during the experimentation phase and **chapt. 5** deals with the analysis of the resulting scenario and specially focuses on the lessons learnt from the experimentation phase.

REIS ROBOTICS – Business and products

REIS ROBOTICS is one of the leading suppliers of robot manipulators and control systems on the European market and is present also in the USA, Asia and South America. Under the main customers of REIS are the automotive supply industries. The research and development departments for mechanical and electrical hardware and control software are located with production and the central application, sales and marketing divisions in Obernburg (Germany). The latter are supported by representation offices in several countries.

REIS faces a tough competition within and especially outside Europe. In addition to the already existing competitors from Japan, other Asian countries are starting to offer cheaper products with comparable quality. Therefore, the robot suppliers are forced to develop innovative products and system solutions to set up successful market strategies. Robots are mainly positioned in the market by three factors: (1) economics (price, costs etc.), (2) performance with reference to the specific applications (technology, quality etc.), (3) after sales (service etc.). The European enterprises have put a lot of effort to influence these factors in a positive way. Many of these efforts lead to successful implementations by the set-up of a proprietary know-how basis, or the ability to fulfil the customer needs in an optimal way.

The main business field of REIS is the development and production of robots and robotic automation systems. The product spectrum of the robots includes standard robot units with vertically articulated, horizontally articulated and linear kinematics with a load capacity in the range between 6 kg and 300 kg. Furthermore REIS provides a complete set of standard peripheral units for the its customers: rotary tables, rotary/tilting tables, linear traversing units, tailstock turning devices and orbital positioners with load capacities in the range between 200 kg and 6300 kg.

In addition to the robotic products REIS also produces trim presses and mould spotting

and try-out presses, that can be delivered in several variants and sizes.
The standard robots and standard peripheral units are put together for many application purposes to complete automation systems. The most important ones are such as: welding, gluing, coating, assembling, handling, interlinking, cutting, palletizing and trimming. The complexity of the installations reaches from simple production cells with e. g. only one robot and a positioning unit for welding applications up to complete production lines with several robots and the necessary peripheral devices for interlinking the robots.
The control of the robots and also of the peripheral devices is achieved by the ROBOTstar V, the in-house developed robot control, that is actually introduced to the market in its 5th generation. The control software that is the subject of the project carried out was originally completely written in assembler and has been ported to the high-level language C in the meantime.

The starting scenario

Experiment context

One of the main business goals of REIS is to maintain and increase its competitiveness on the market by providing the robots with control systems that can be cost- and time-efficiently extended and modified. An improved extendibility and modifiability of the control system has also to enable REIS to follow the newest trends in control technology promptly and provide up-to-date control systems. OO SE methods and tools are promising approaches to achieve these requirements.

Technical objectives
Based on the above considerations the following technical objectives of EMESO were defined:

- investigate and select an appropriate OO design approach for software for embedded systems,
- select an appropriate toolset supporting OO design,
- test the selected method and toolset for the design and implementation of a restricted number of software modules in the robot control,
- test extendibility and modifiability of the OO modules and compare the results with the same features of the modules developed based on classical approaches.

People related objectives
One of the objectives of EMESO is to enable that the people in REIS get trained in the application of the OO methods and tools and get confidence in these methods and tools and to eliminate reluctance towards the application of OO approaches. The objective is also to define an optimal organizational structure for OO design and implementation at REIS.

Long-term objectives
The objective of EMESO is to provide clear assessment and judgement on efficiency and both technical and commercial benefits of the selected OO methods and tools for REIS. This will serve as a basis for defining the detailed migration path to establish the OO methodology in the whole company.

Relation to the business goals

The stated technical and people related objectives will directly contribute to the achievement of the following business goals:

- reduce the efforts for extensions, modifications and maintenance of software for the REIS robot control,
- reduce the time required for extensions and modifications of software modules (reduction of time-to-market),
- ensure quality of the extensions and modifications of software modules and to reduce the number of errors identified in the integration tests.

Company Context

Currents Strengths and Weaknesses

The main **strengths** of the current status of the SE practice at REIS are:

- REIS is strong in the requirements analysis of customer needs due to its good understanding of the customer's processes as a system vendor,
- organization is appropriate (e. g. clear definition of responsibilities etc.),
- project management is well performed.
- In reference to the stated company business objectives, the main **weak points** of the present SE process are:
- the SE process is based on classical functional approaches that constrain strongly extendibility and modifiability of the software; thus, the work for extensions and modifications is very time consuming,
- maintenance is asking for a lot of efforts,
- tests and integration are not sufficiently controllable,
- the application of metrics is not satisfactory (only efforts and time spent are normally measured).

Technical environment

The software of the REIS robot control is implemented in C, and some core elements are still written in assembler. The robot control is VME based using cross compiler tools on UNIX platforms. Tools for automatic generation of code are only used in rare cases.

People related aspects

REIS staff members in software development are highly experienced in the development of software for robot control systems as well as in the modification of such systems to accommodate specific customer requirements. There is, however, little experience with the application of modern SE methods and tools. This is leading to a certain reluctance towards the application of OO methods and tools. The main fear is that such methods may require too high computation time and memory resources and the given real-time requirements might not be reachable with the actual hardware configuration.

Quality assurance aspects

The quality of software, being one of the main competitive feature of the REIS products, is achieved by intensive testing the system in the integration phase. However, the efforts for these tests are very high since the number of errors discovered in this phase is rather high and not well controllable. This seems to be a consequence of the way how extensions and modifications are performed.

Required corrective actions

Based on the above analysis, it can be concluded that the main corrective actions required at REIS are:

- improvement of the design and implementation phase by more appropriate methodologies which enable easier extensions and modifications,
- improvements in the technical environment which will enable a more efficient development in the design and implementation phase,
- the methodology should enable REIS to ensure the high quality standards in a more efficient way than presently,
- improvement of the skills and experience of the staff with reference to modern SE methods and tools,
- improvement in the configuration management system,
- introduction of appropriate metrics to control the software process better.

Baseline project context

The baseline project for EMESO is the development of a new extended version of the existing REIS robot control. The control is implemented in C (85 %), but some core elements are still written in assembler (15 %). The robot control is VME based using cross compiler tools on UNIX platforms.

The baseline project essential for the success of REIS in the near future. The baseline project has been started in June 1998 and will according to the plans last about 14 months. The new version of the robot control is scheduled to be ready in the end of 1999 in order to meet requirements of certain important customers.

Since this baseline project addresses extensions and modifications of the existing software modules, it is very appropriate to test advantages with reference to extendibility and modifiability of software. EMESO started in June 1998 and fits well into the schedule of the baseline project. Since this has a high relevance for REIS, the successful application of the OO methodology provides a good basis for the decision on the investment in such an advanced methodology.

Implementation and improvement actions

Phases of the experiment

In order to achieve the planned objectives and in reference to the identified ”weak points” of the REIS software development process, a detailed analysis of the SE process at REIS was performed. The main tasks were to analyze the existing SE process in detail, to check the new SE methodologies for the REIS software development process and to select and test OO methods and tools for design and implementation of the software modules for the robot control system.

Thus, the technical work of EMESO was divided into 3 main phases: **Phase 1** covers the analysis of the state-of-the-art and the preparation of methods and tools for the experiment. **Phase 2** contains the work of the experimentation phase, i. e. the design, the prototypical implementation of selected parts and the testing of the results of the modifications. The following **phase 3** contains the tasks that use the results of the experiments for planning the future activities to introduce the OO methods to the REIS control software. Additional work concerns documentation and dissemination activities of project results. In detail the phases of the project contain the following tasks:

Phase 1

Different OO methods and tools concerning the specific requirements and constraints of real-time software development are evaluated. They have to support all relevant design views for real-time systems. Therefore, the features of different software tools that support the software development process for real-time embedded systems are assessed with regard to:

- operational costs,
- facilities to support design and implementation,
- flexibility and capability to handle dynamic aspects
- minimize requirements with reference to computation time and memory resources of the target systems,
- investment costs,
- ability to be compliant with different company-internal SE standards,
- ability to support the transition between the different phases.

The baseline project is analyzed to identify which of the extensions of the existing control, can be taken to compare the efforts for the OO approach. The OO kernel of the experimental system is defined as well. Specific attention is paid to the facts, that the absolute size of the selected modules is as small as possible, but the complexity of the interaction of the module with the control is typical for extensions of the REIS control. A result is the selection of modules for the extension of the existing control with a typical complexity of the interaction with the control kernel.

Appropriate metrics are selected to get quantitative measures. The measurements concern the time necessary for development of the extensions as well as the comparison of the efforts for integrating additional modules in the existing control and the test kernel. Different metrics are considered in order to measure which metrics are most useful. An essential measurement is the run time performance of the generated code. Further metric aspects to be addressed are assessment aspects with reference to the level of the encapsulation of the modules and the complexity of decomposition structure of the software architecture. The measurement values achieved by the application of metrics are compared with the measured values achieved by direct human judgement. The corresponding deviations are measured and evaluated.

Phase 2

The OO control kernel is designed and implemented based on the selected minimal requirements defined in phase 1. Because the control kernel provides only a test architecture for extensions, it is implemented with minimum efforts. The kernel is tested in order to ensure that it properly encapsulates all relevant features for testing the efficiency of the OO approach.

The design and implementation of the selected extensions and the documentation of the needed efforts are objectives of this phase. The needed efforts are documented as well as the efforts for the corresponding prototypical module implementation, integration and tests in the baseline project. Design, prototypical implementation and tests in EMESO are carried out according to the selected OO methods.

A training of the several staff members on OO methods and tools is carried out. Different organizational aspects are also considered (e. g. organization of team work and ways of co-operation among developers etc.).

The measurements according to the metrics selected in phase 1 are also carried out.

The required efforts, the quality of the code and the run-time behaviour and necessary resources of the experimental design and implementation are compared to those in the baseline project. The resulting metrics from phase 1 are chosen for measuring the improvement of the code quality and the run-time behaviour of the OO software.

Phase 3
The analysis of the improvements in the REIS software process is carried out, as well as the analysis of the lessons learnt during the experimentation phase. The specific experience in application of OO methods for real-time robotic control software is formulated in the form of guidelines for specification, design, implementation and testing at REIS. These guidelines are initially prepared in the scope of phase 1, but they will be redefined at the end of EMESO to be used in further projects. The consequences of the results of the project upon the different aspects of the software process at REIS are analyzed.

Dissemination and documentation
Internal disseminations are carried out according to the planned activities. The technical staff level in software and hardware development and in the production as well as the management level will be addressed by these disseminations. The knowledge concerning the application of OO specification and design methods as well as the aspects relevant for the management of OO based software projects are disseminated to the remaining development personnel at REIS.

Consultancy during the experiment

In order to use its resources optimally REIS subcontracted ATB – the Institute for **A**pplied System **T**echnology, **B**remen – to support the selection and the introduction of OO methods and tools at REIS. Due to relatively low experience of REIS staff members with OO methodologies, such an approach was necessary to ensure the appropriate execution of the experiment. In reference to its expertise ATB provides support in the following aspects:

- detailed selection of appropriate methods and toolsets for OO design and its preparation to accommodate the specific needs of REIS,
- application of the OO methodology during the experimentation phase,
- selection of metrics and execution of the measurement of the project results,
- suggest special OO realization methods for integration in the REIS robot control system,
- detailed planning of the improvement process at REIS.

Results and lessons learnt

Results

Technical impact
The analysis of the existing software structure of the control system and the selection of the correct methods and system parts to be redesigned by OO methods has been the key part of the project that has been carried out very carefully to obtain a good basis for the OO redesign of the complete software.
The main result from the analysis phase of the robot control system is the selection of

one key module of the system: the interpreter, that is responsible for analyzing and interpreting robot programs and for controlling the robot movement. This module forms one centre of the control software and has interfaces to the main parts of the control. Well defined interfaces for the interpreter are the basis for a later translation of further modules of the control software into OO structures and C++. The basic structure of the robot control is shown if **fig. Dr.2**.

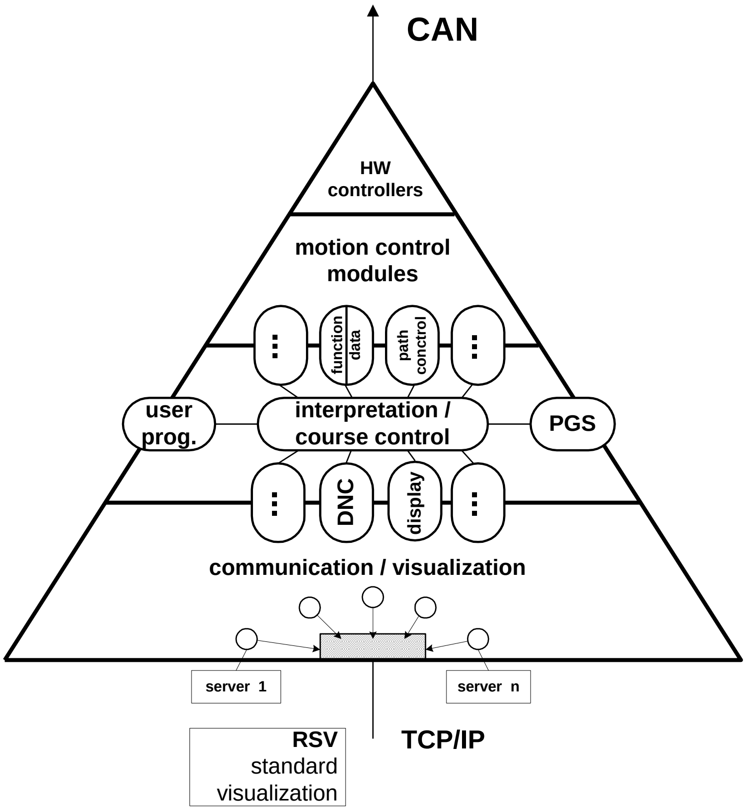


Fig. Dr.2. *Basic architecture of the robot control.*

The selection of the interpreter for being redesigned in an OO way has several consequences for the activities carried out in EMESO: The combination of OO designed and classically designed software modules in one control is not reasonable. This condition is surely valid for the release version of a robot control. For the experimental system for prototypical tests of OO designed control parts, a mixed structure with OO and non OO modules is acceptable. For testing the interpreter it is necessary to have a more complete platform that also contains modules still written in C. These modules are necessary to keep the experimental system running.

Generally, a lot of effort also has to be spent on the dynamic memory management, that is a critical part in C++-written software. But especially for embedded systems the dynamic memory handling has to be designed and created very carefully due to the limitation of resources in those systems and due to the long run time of embedded systems. A memory fragmentation or leak is not acceptable because this will lead to a complete system crash, that in a running automation system must never happen.

Also some other features that are convenient in C++ have to be taken into account very carefully to deal with the conditions in embedded systems (e. g. recursive structures), because these methods are memory consumptive and one has to manage with limited resources in embedded systems. In a first approach those methods should be avoided at all. Only in rare cases one should make any exceptions of this.

The automatic creation of variables in a wider range in C++ is also a point that has to be paid attention to. I. e., that all of the constructors and destructors have to be designed very carefully so that they don't work memory and time wasting.

The evaluation of the applicability of OO tools with a version of Rational Rose at ATB has shown that the tool is providing no sufficient support for the design of the software needed by REIS. The problem REIS is faced with is the need for a detailed algorithmic specification or an analysis of the effects of different language constructs of C++. This, however, cannot be done on the abstract level which the tools are mainly supporting. Interfaces and class hierarchies are defined to structure the overall system, but all are affected by details of the implementation. Therefore, the structure of the overall system is dominated by things which may not be easily visible in the tool and, on the other hand, the views supported by the tool may not be very helpful for the design. Thus, Rational Rose would be a too expensive tool to just draw class diagrams. A tool which is more helpful for the development is a profiler allowing for the measurement of the used CPU time and memory consumption of different code sections.

The selection of the metrics is faced with the principal problem of comparing OO code with non-OO code. On the one hand it is very obvious of how an OO design approach can improve the structure of the system. Once it has been proven that the implementation fulfils the requirements of an embedded system, it not only allows for an easier understanding but also a better overview. On the other hand, a quantification of such an improvement by certain characteristics of the code should be possible. Some algorithmic parts of the code are not expected to be changed much, because they only use internal data and their functionality is not affected by the restructuring. However, these parts of the code for the interaction between modules will be greatly simplified due to the layering of the complexity with the help of interface functions to data structures. Such simplifications should be visible by means of the metrics 'Cyclomatic Complexity' and 'Average number of segments per path'.

Additionally, another improvement will be the de-coupling of different modules. The present code is characterised by an extensive use of global variables. Although efforts were spent to change this situation, the range of effects in the case of changes in the code at one place is still too high. This is planned to be considerably changed due to a functional interface to modules realized by class methods providing an interface on a higher level of abstraction. This will be reflected in the 'Total number of global variables used' and in the 'Average number of modules sharing a data type'. The number of modules sharing a data type represents the number of modules which have to be adapted in case of a change. Therefore, the improvement of the OO design should be indicated by measuring these values.

Other special OO metrics are not expected to be helpful in comparing the C++ code due to the special nature of embedded systems, on the one hand and due to the algorithmic structure of the robot controller on the other hand. Based on these considerations it was decided to carry out the following measurements:

- Cyclomatic Complexity,
- Average number of segments per path,
- Total number of global variables used,

- Average number of modules sharing a data type.

Business impact

Up to now there are no direct impacts from the EMESO project perceptible, because those changes of the SE process normally have medium-term or long-term effects. But the results of EMESO, e. g. a shorter time-to-market and a better portability of the control software and also a better maintainability and reuse of the OO structured software will surely have positive effects in the future. Those impacts will take effect when a new version of the control is introduced to the market.

At the beginning of the project we defined measurable quantitative objectives. However, these goals can only be measured at the end of the project. Find following an indicative table to demonstrate the intended measurement of the goals:

Item	Goal
Reduced costs for extensions, modifications and maintenance	20 %
Reduced time-to-market for modifications	30 %
Reductions of errors in integration tests	20 %

Organizational impact

No definite organizational changes have not been made during EMESO up to now. But already now there are possible changes in the organization of the software development process and the software development department recognizable. The persons involved in the project will obtain a stronger position with more responsibility because they get an improved knowledge through the training and because of the work with new methods and tools. In addition, there is a need for a more intensive information exchange between all people concerned.

Culture impact

In the initial phase of the project the additional work concerning the necessary more detailed definition and designing overhead as well as the more rigorous methods led to a slight negative attitude towards the new methods. But after the training and after recognizing the positive effects of an OO approach the attitude was no longer negative. The training of the people in OO design and programming led to an improved co-operation between the developers to transmit the knowledge to the persons not involved in the training. Due to the use of OO structures in future there will thus be necessary a closer teamwork in general.

Skills impact

The training of the people in OO design and programming leads to a better knowledge in OO structures for designing and programming. But in addition to this the training leads to an OO way of thinking in general. This means, that the SE process generally will be affected by the improved OO way of thinking of the people. Besides this, the use of new SE tools leads to a wider knowledge in CASE tools. A wider use of tools rather than achieving things by do-it-yourself-methods could also be a positive impact from the project.

Lessons learnt

Technological point of view

Besides the direct technical results there have been learnt some lessons from EMESO that will have principal effect to the software development process. These refer to the following points:

- Importance of good data structures and well defined interfaces,
- Importance of education and training to make the people able to use more sophisticated development methods and tools to improve the software structures,
- Importance of modularity of the code to be able to affect things through changes in one module rather than through changes spread over the whole code,
- Importance of code reuse through modularity of the modules,
- Importance of internal disseminations to improve the knowledge and information exchange between the people,
- Better knowledge about the actual software through the very detailed analysis carried out in EMESO,
- Better knowledge about how the software should be structured.

Business point of view

Some of the lessons learnt concerning the business point of view are the following:

- Importance of platform independence of the control software to have better choices for future developments towards standardization and cost reduction,
- Importance of easy achieving big software changes by minimum efforts through modularity of the software,
- Importance of software reuse that leads to a reduced use of memory, a reduced amount of testing, service and maintenance. Software reuse leads to easier introducing of new functionalities without big implementing efforts.
- Importance of investing in education and training to be able to use more sophisticated methods and tools which lead to shorter development cycles in future.

Strengths and weaknesses of the experiment

From the actual point of view there exist some factors that point out principal strengths and weaknesses. The identified **strengths** are as follows:

- The code structure of the control software is analyzed in an very detailed way.
- A plan for the principle structure of the software is obtained.
- A plan for an improved data structure and interface definition is obtained.
- The efforts of the people involved in the project are recognized by the management.
- The identified **weaknesses** refer to the following points:
- Identify the big efforts necessary to achieve the recognized improvement actions.
- Identify that due to the heterogeneity of the software structure developed over the years the selection of appropriate tools is difficult; the suggested tools cannot achieve the performance as expected.
- The necessary preparation of the staff before starting the tasks of the project.

Conclusions

New SE methodologies are necessary in the REIS SE procedure to achieve the main objectives. EMESO gives the opportunity to carry out a relative small experiment besides the main actions taking place in the SE practice to test, to check and to measure special methods. This makes it easier to get into new technologies than it would be without the project.

In future the results of EMESO have to be inspected accurately to make the right decisions how the found conclusions can be applied in general.

In future similar changes that have a general effect to the software structure or the SE process could be realized by a similar experiment after the thoroughly positive results from EMESO. Depending on the plans of the European Community to fund software activities in a similar way than today, EMESO encourages REIS to take into account to carry out further projects in a similar situation in the future.

References

The best information on *Object Orientation* can be found on the internet under http://www.rhein-neckar.de/~cetus/software.html. Further selected references are:

[1] Burkhard, Rainer: “UML-Unified Modelling Language, Objektorientierte Modellierung für die Praxis”. Addison-Wesley Publishing Company, 1997

[2] Fowler, Martin: “UML Distilled”. Addison-Wesley Publishing Company, 1997

[3] Gamma, Erich et al.: “Design Patterns”. Addison-Wesley Professional Computing Series, 1994

[4] Rüping, Andreas: “Software-Entwicklung mit objektorientierten Frameworks”. Ph.D. Thesis, University of Karlsruhe, Shaker Verlag Aachen, February 1997

[5] Shaw, Mary, Garlan, David: “Software Architecture”. Prentice-Hall Inc., New Jersey, 1996

Session 14
SPI and Testing II

Chairman
Yingxu Wang
IVF, Gothenburg, Sweden

Improvement of development process through enhanced test procedure & change request management - ImproveTCR

Marek Blaszczyk
DAKOSY GmbH

*Cremon 9
20457 HAMBURG
Germany
http://www.dakosy.de*

Introduction

Many companies which have chosen to introduce object-oriented technology have started with design and implementation and then gone on to expand the technology to include analysis. Other companies have tended to adopt the opposite approach. What many companies have in common, however, is that they have ended the process of introducing the object-oriented technology after implementation, i.e. after an object-oriented programming language has been introduced. The substantial effort involved, combined with the frequent lack of success, means that the test process remains unchanged in its original form. Systematic use of the object-oriented technology requires the test process to be adapted to the new object-oriented software development process.

This report describes our experience of changing over from the classic test (V-model) to the object-oriented test process.

This experience is based on the results we have achieved at the half-way stage in our PIE project *„Improvement of development process through enhanced test procedures and change request management “*- **ImproveTCR - 27352**. The **ImproveTCR** project is funded by the Commission of the European Communities (CEC) as a Process Improvement Experiment (PIE) under the ESSI programme.

These results are especially significant for firms which have not yet adapted their test process to object-oriented requirements or which are looking for new ideas in this area to increase quality and productivity. The decision to view testing not as the last activity in the software development process but as a parallel process which supports and accompanies software development on a long-term basis, provides useful information about both the resulting product and the development process. This is necessary for everyone who is interested in continuously improving the process.

Starting scenario

Business motivation

The Port of Hamburg is a classic hub of intermodal transport, handling thousands of containers each day (1997: 3.34 million TEU). For the transport process to work properly, information must be transmitted as far as possible ahead of the actual transport and be accessible at all times. This requires rapid communication between all the firms involved in the transport process.

The function of DAKOSY is to act as a "data junction" linking together all the companies and institutions involved in the handling processes, by means of EDI (Electronic Data Interchange). All transport documents and orders can be exchanged via a wide variety of EDI services and managed with many advanced application systems.

DAKOSY’s main areas of business are:

1. Operating the data communication system created by DAKOSY for the Port of Hamburg with 99.8% availability.
2. Carrying out all work which is necessary in the field of data processing with a view to ensuring the smooth functioning of the Port of Hamburg.
3. Providing support and advice to companies in the field of data processing.

DAKOSY’s range of services includes **software development, computer centre services** and **DP consultancy** for the port transport sector. With every service which DAKOSY provides now and will provide in the future in the port area, the top priorities are system reliability, speed and reliability of information processing.

One part of achieving the functionality of the port is the project HABIS II/2 (**HA**fenbahn **B**etriebs- und **I**nformations **S**ystem). This software application, which is being developed for the City State of Hamburg, covers approx. 23 person-years and is in the middle of its production cycle. The application is of central importance for goods transport in the port of Hamburg. The functions of order planning, disposition and implementation together with order monitoring are the main attributes of the system. This project is also the Baseline Project for the PIE.

The other motivation for the PIE project comes from the business environment: increasing competition, constantly increasing demands on the reliability of DAKOSY systems on account of the economic importance of the applications for the user.

Experiment context

The project objectives which we aim to meet relate to the Baseline Project but are also expected to become established within the company as a whole. This task is not difficult provided that the Baseline Project reflects the development environment of the entire company. In our specific case we deal with a variety of different development environments such as COBOL and RPG developments on the AS/400, Microsoft Visual C++, PowerBuilder, Visual Basic on Windows NT, and JAVA on both platforms (AS/400, Windows NT, and Internet). The Baseline Project shows only part (MSVC++) of the spectrum covered within the company. For this reason our experiment not only relates to the chosen process organisation, its methodology and applied technology, but also gains the additional dimension of company validity. Our process is defined in such a way that it is independent of the tools and systems used.
The tested product/part of a product is termed **test object** in Figure 1. The test method, which includes the test procedure, is determined in relation to the developmental phase. The test procedure considers relevant test techniques. Supporting tools are allocated to the test procedures.

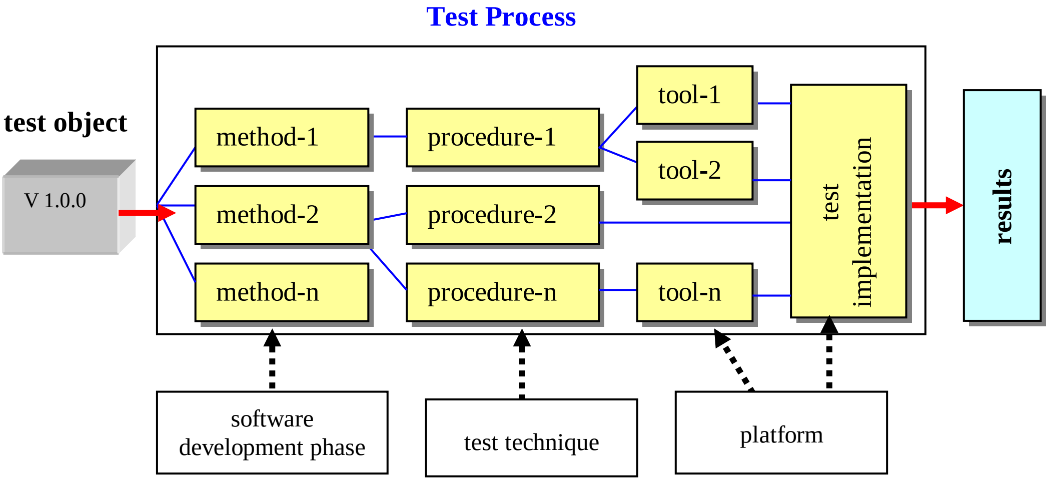


Fig. MB.1: Test process

A further main aspect of our project is that the test process follows the software production process. The quality of a complex system is determined by the quality of its individual components. We therefore do not regard the test process as the final stage of the software development process, but as a parallel process closely connected to the software development process (Figure 2).

The test process must provide an appropriate procedure for each test object (independent of the particular developmental phase from analysis to implementation). This includes the consideration of all available information on any one test object, e.g. UseCase models, class models, sequence diagrams and collaborations diagrams in order to minimise effort.

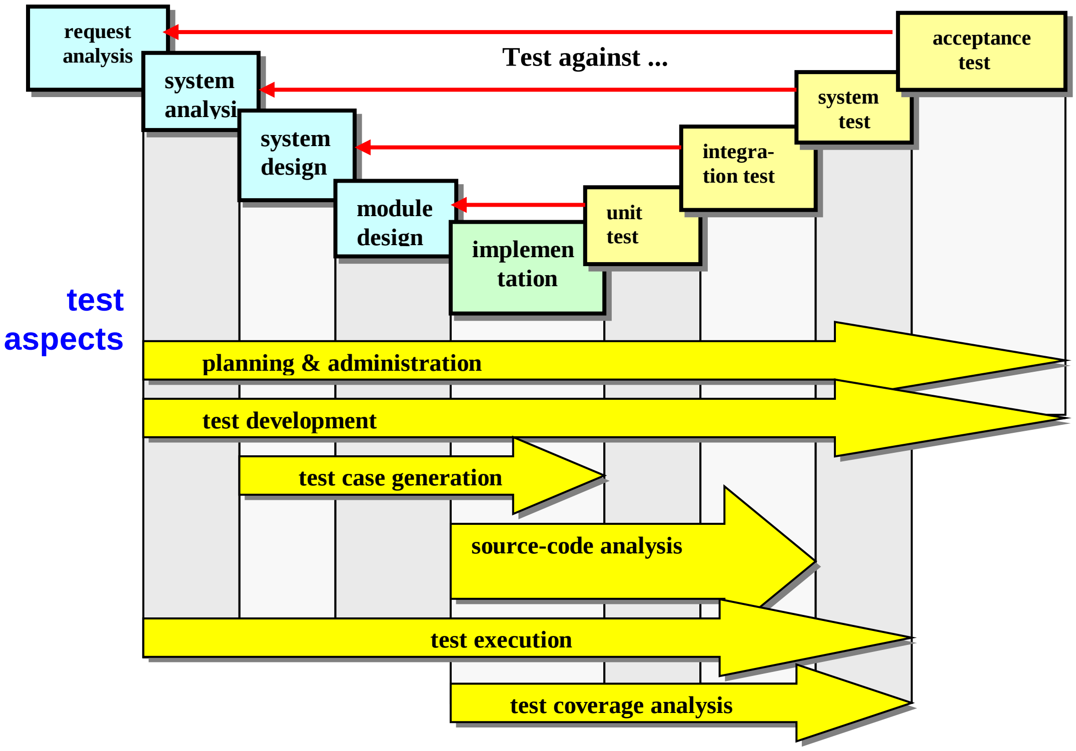


Fig. MB.2: Testing in the context of software development

This approach allows the following aspects of the previously defined software process to be examined:

- Are the UseCases defined also specified with sufficient clarity for the test process?
- Can appropriate test cases be derived from the collaboration, sequence and state diagrams?
- Does the system architecture provide appropriate information which can be used to define adequate test procedures?

A further important aspect of the experiment is the expansion of testing to include the analysis and design phase. In other words, checklists are to be used to ensure that the specified information and design models are recorded completely and systematically. The later a design fault is discovered, the more expensive it is to remedy it. This is very important in terms of efficiency, productivity and quality of the product and process.

With this approach we aim to achieve a higher degree of efficiency with regard to technology and business.

Baseline application

The Baseline application is a prototype implementation of the most important operating masks of incoming and outgoing journeys, initially with a low degree of functionality. The prototype is based on selected specialised classes and their link to the system database. At the same time, it is also used to evaluate both the system architecture design and the software development process.

The HABIS II/2 prototype 1.7 (*HABIS II/2 represents the further DP development of the HABIS II/1 system*) has the following distinguishing features:

1. It produces a visualisation of transport orders. Transport orders from HABIS II/1 are displayed and broken down into planning units.
2. It produces a reading interface to the HABIS II/1 system.
3. Data from the HABIS II/1 System are incorporated into the HABIS II/2 database.
4. Train journeys (arrivals and departures) in the HABIS II/1 system are incorporated into the order scope of the HABIS II/2 system.

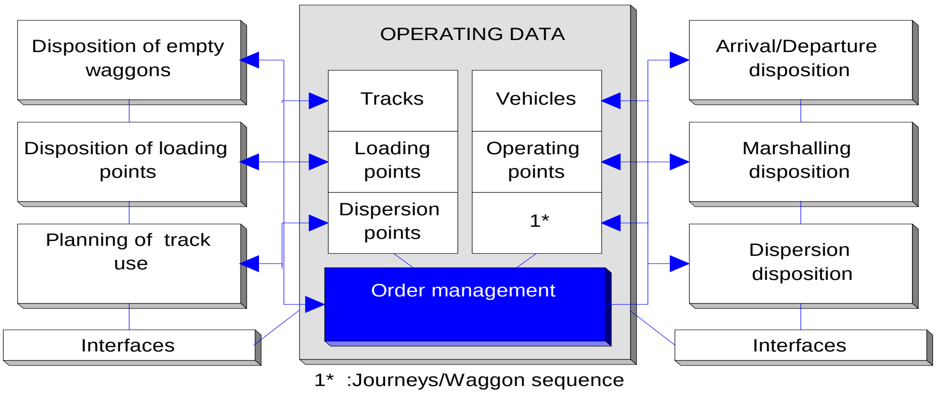


Fig.MB.3: Application architecture

Status before the experiment

The client/server development process, based on object-oriented technology, which DAKOSY introduced in its offices in 1996 requires a different software development procedure than the host applications developed hitherto on AS/400: an iterative and incremental procedure. The software development process includes four individual phases: Analysis, Design, Implementation and Test.

Early results from the analysis and design phase are tested using prototypes. With the knowledge obtained from these prototypes the analysis phase is run through again. This makes it possible to:

- Recognise risk factors early, and where appropriate eliminate them.
- Integrate into the modelling process any new knowledge gained through implementation.
- Involve the user in the development process.

The result is a cycle which is repeated a number of times within a project. These steps are controlled and monitored by project and quality management. DAKOSY has been certified in accordance with the **DIN ISO 9001** quality standard since 1995.

In spite of the cyclical procedure, the development process is not without risks. This part of the process is too dependent on the developer inasmuch as he is obliged to carry out all the steps as they are laid down in the QM handbook. There is a lack of support for him in the form of methods and procedures which the developer can supply with information such as errors found, type of error, release processed. The process in which information is gathered should automatically lead on to further processing steps, such as: drawing-up of test documentation, error correction process, generation of new release, reverse engineering and model correction and so forth.

This would ensure that the processes interlock automatically and that nothing can be confused and forgotten.

The multi-layer client/server architecture which has been introduced, and in which the layers in the network can be distributed in any way, increases the risks of software development. The result is that many more factors have to be considered during implementation and testing. The developer is dependent on his own experience and creativity to be able to formulate as many aspects and test constellations as possible. The result of the phase is then reflected in the number of errors detected in the product delivered. The process needs improvement in this area.

Plans and the expected outcomes

Project objectives

The PIE project in the areas of **testing** and **change request management** is intended to bring medium and long-term business improvements in a number of areas:

- Increase productivity by 20 %
- Increase quality by 50 %

In addition there are other, not directly measurable, objectives: increased product reliability, customer satisfaction, development staff motivation and awareness of quality management.

From a technical point of view, the objectives of the project are as follows:

- Reduce software error rate by 50%
- Reduce testing effort by 30 %
- Reduce correction time by 50 %
- Increase process transparency

In the medium to long term the following effects are expected:

- Efficiency and transparency of the test process and the change request process within the company. The individual activities are more strongly triggered by the process itself than by a member of staff because of a clear definition and a tool-supported completion. In this way the processes reach a higher level of human independence and a consistent translation.
- Customer satisfaction. Customers are already following the process improvement with increasing interest.

Phases of the experiment

Workpackage overview

Ref.	Activity	Phase	State
WP0	Project Management		In progress
WP1	Co-operation with other ESSI Projects		In progress
WP2	Process Assessment Part I – Begin	I. Initialisation	Concluded
WP3	Process & Organisation Definition		Concluded
WP4	Process Improvements Metrics	II. Process Measurements	Concluded
WP5	Technology	III. Evaluation of Technology	Concluded
Milestone I			
WP6	Improvement of Test	IV. Improvement Part **I**	Concluded
Milestone II			
WP7	Improvement of Change Req. Proc.	V. Improvement Part II	In progress
Milestone III			
WP8	Product of the Improvement		Future
WP9	Result Evaluation	VI. Result of Evaluation	Future
WP10	Dissemination Activities		In progress

Table 1. Workpackage activities

Workpackage **WP2** was assessed with the assistance of the management consultancy DTK GmbH. The entire software production process was analysed with the help of the

Bootcheck tool in version 3.0. The SW process **(WP2)** evaluation [3] combined with the knowledge [1][4][5][10][14][15] and experience gained from other projects formed the basis for the test specification and change request process **(WP3)**. The result of the specification is the “process definition” [2] which describes the “improvement process” and the processes “testing “ and “change request” and their size within the experiment.

The metrics which are relevant for the project were worked out using GQM method [9] in **WP4**. With the aid of these metrics it will be possible to adequately evaluate the development, i.e. the improvement, in the project [6][7][13]. The metrics form the basis for the analysis of the process and they are summarised in the document "Process Improvement Report" [11].

The basis for the tool evaluation **(WP5)** was mainly provided by the process definition **(WP3)** and the process and product metrics **(WP4).** The pre-selection was made on the basis of the “Ovum-Report” “ Software Testing Tools” [12]. The pre-selected tools, which best supported the defined process and conformed to the relevant commercial, technical and technological aspects, were examined and evaluated. A comprehensive tool evaluation report has been prepared.

Table 1 gives an overview of the test tools used and the main areas where they were employed.

Test tool	Manufacturer	Main area of use	Type of test
Cantata++	IPL Ltd.	Developer tests – White-Box Tests	Class test Cluster test
SQA TeamTest	RATIONAL GmbH	Technical Test – Black-Box Tests	System test
Logiscope	VERILOG	Source-Code Analysis, Metrics Test coverage – Black-Box Tests	System test
Purify	RATIONAL GmbH	Developer tests – Memory error	Integration test System test
PureCoverage	RATIONAL GmbH	Developer test – Test coverage	Integration test

Table 2. Test tools used

This workpackage forms a critical aspect of the project. A wrong tool selection would not only result in waste of resources, but it will also jeopardise the process by preventing the achievement of the targets identified. Consequently, the tools were scrutinised and tested where evaluation licences were available.

Individual analysis and design models, black- and white-box tests have begun (**WP6)**. The activities of the workpackage were finished on 30.06.99. The activities of **WP8**, which ensure distribution and stabilisation of the objectives achieved, can be carried out subsequently. A second evaluation using the “bootstrap method” will be conducted **(WP9)** at the end of project activities and following an evaluation of the process and product data of the two process cycles. This will assess whether the project objectives have been achieved.

Implementation of the improvement action

Process sequence

The initialisation, process definition and tool evaluation phase was followed in the period from 01.02.99 - 30.06.99 by the first cycle of the test process (WP6). The test process was divided into two cycles. The main emphasis of Cycle 1 is the implementation of the test process as an automatically running process. This meant that it had to be clearly defined, responsibilities had to be assigned, selected test tools integrated and product and process data measured.

In Figure 4 this corresponds to Curves 1-4. After the process has been evaluated, the automatic stage follows. The initial difficulties must be eliminated before this stage (Curve 5). This is the sequence of Cycle II. Its main emphasis includes the intensive collection of process and product data and verification of the process in respect of use throughout the company

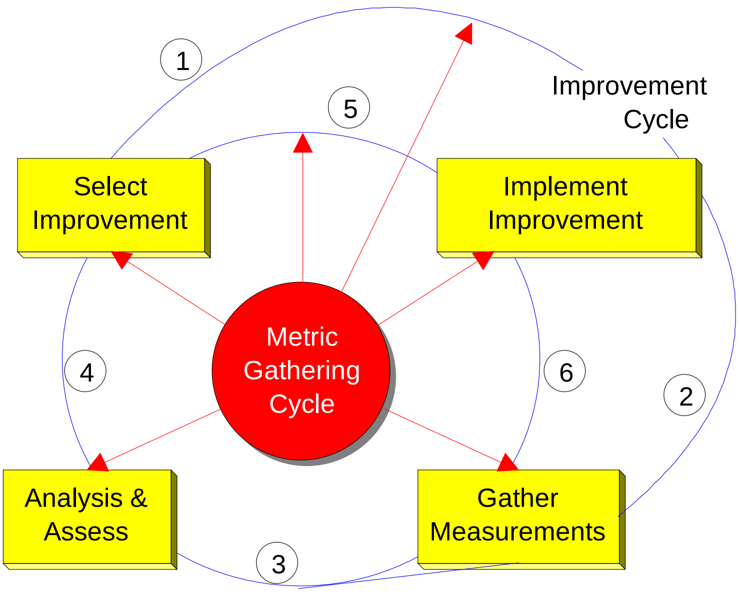


Fig. MB.4: Process sequence

In parallel to the test process in Cycle II, change request management is introduced into the process. Here too, it is to be expected that as a result of the build-up activities, the change request process will adopt first Curve 1 and later Curve5.
The shape of Curve 1 in comparison to Curve 5 is intended to symbolise the additional effort that is incurred during the build-up phase
This includes:

1. Problems installing or using the test tools and their errors

2. Learning how to use the test tools in practice
3. Effort involved in integrating the test process with configuration and project management
4. Definition of various documents and templates
5. Co-ordination effort on the part of those involved in the process
6. Corrections in the process sequence

There are thus technological, organisational and methodological factors that can have a negative influence on the process in the build-up phase. For this reason, the data obtained have a different significance from those obtained in the course of Curve 5.

Improvement actions

To improve the development process within the Baseline Project, the following methods and procedures have been established:

- **Test planning and test case** execution have been established. We have the possibility to manage the whole test process with the identification of test requirement, the planning of test cases and test procedures. With this set of test cases we have been able to integrate in our test process a regression test functionality.
- **Integration of design specifications into the test process.** The specifications which were established in the design phase can be adopted as requirements in the test process.
- **White-box tests** – specification and procedure of the different test types are defined : class and cluster tests.
- **Black-box tests** – specification and procedure of the different test types are defined : GUI test (GUI-Graphic User Interface), performance test, memory test, integration test, system test, acceptance test.
- **Effectiveness of the testing** is established on the basis of the test coverage.
- **Introduction of metrics** for products and processes which facilitate fast analysis of the test process and product and therefore minimise risks. Metrics analysis enables early recognition of negative trends.
- **Establishing the metrics of products and quality.** In order to be able to assess the effectiveness of the testing, one requires information about the complexity of a product, e.g. depth of inheritance, number of children, coupling between objects, coupling between classes etc., to put them in the context of established results (e.g. number errors, test effort).
- **An acquired knowledge database** was implemented to store results and experiences. This database can be accessed by any member of staff at any time.
- **Checklists for analysis specifications and design models** were developed and implemented successfully.

Measured Results, Impact and the Lessons Learned

The measured results

Errors

Within the scope of the Baseline Project, the new test procedures were used to develop a prototype of the future system. The incremental and evolutionary software development procedure allowed a number of prototypes of the system to be constructed.

Prototype–Version 1.7.1 - with a basic functionality
Prototype–Version 1.7.2 - expansion of basic functionality
Prototype–Version 1.7.3 - further expansion of functionality

Versions 1.7.2 and 1.7.3 incorporate the error correction from the previous version. Table 2 shows the results of the various tests.

Number of test objects per test type			**Number of test cases**		**Number of deviations**
			Main cases	**Derived test cases**	
Class test		3	18	4,912	**5**
System test					**42**
	Technical requirements	53	210		
	GUI	23			
	Performance	13	13		
	Memory	1	210		
	Function coverage	1	210		
	Installation	1	1		

Table 3. Overview of test cases

Within the scope of the class tests, 3 classes which have a very high degree of replicability were tested. 3 test cases were established here. The formation of the equivalence classes generated a large number of subtest cases which were tested **-4,912** in total. 5 errors were discovered during the tests.

A total of 42 errors/deviations were found during the system test. This number of errors represents the sum of the errors from all three prototypes. Figure 5 shows the number of errors for each version. It is noticeable that the errors mainly occurred in the earlier versions. This is a positive effect of early testing.

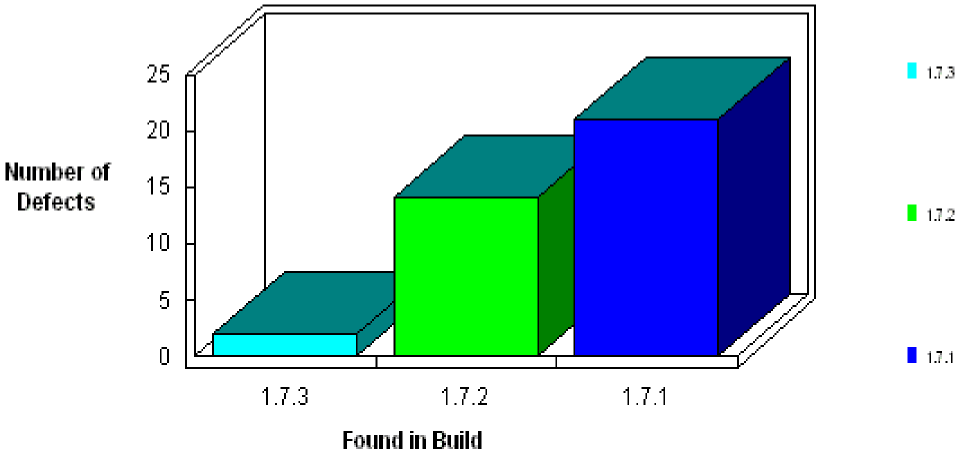


Fig. MB.5: Application defects distribution

Other Metrics

The scope and key data for the applications are shown in Tables 3 -6. These data are very important for the evaluation of the productivity of the process and the quality of the products. The basic metrics for the scope are the number of Statements (STMT) and not the number of Lines of Code (LOC). Statements are defined as an executable instruction independent of the number of Lines of Code.

APPLICATION VOLUME :	NUMBER
Number of classes	**220**
Number of methods – Private	**1,980**
Number of methods – Protected	**113**
Number of methods – Public	**460**
Total – Methods	**2,553**
Statements and Commentaries :	
Number of Statements	**10,063**
Lines of commentary before class definition	**3,940**
Lines of commentary in classes	**826**
Total	**14,829**

Table 4. Scope of application

Class data :	Value	Class name
Highest complexity of a class (total VGs)	**208**	CbasCtlListCtrl
Highest number of derived classes	**7**	CpersistentObjectDbzo
Highest number of dependent methods	**171**	CMSFlexGrid
Highest coupling (max. number of connections)	**312**	ClehFrtDbzo

Table 5. Maximum values of the application

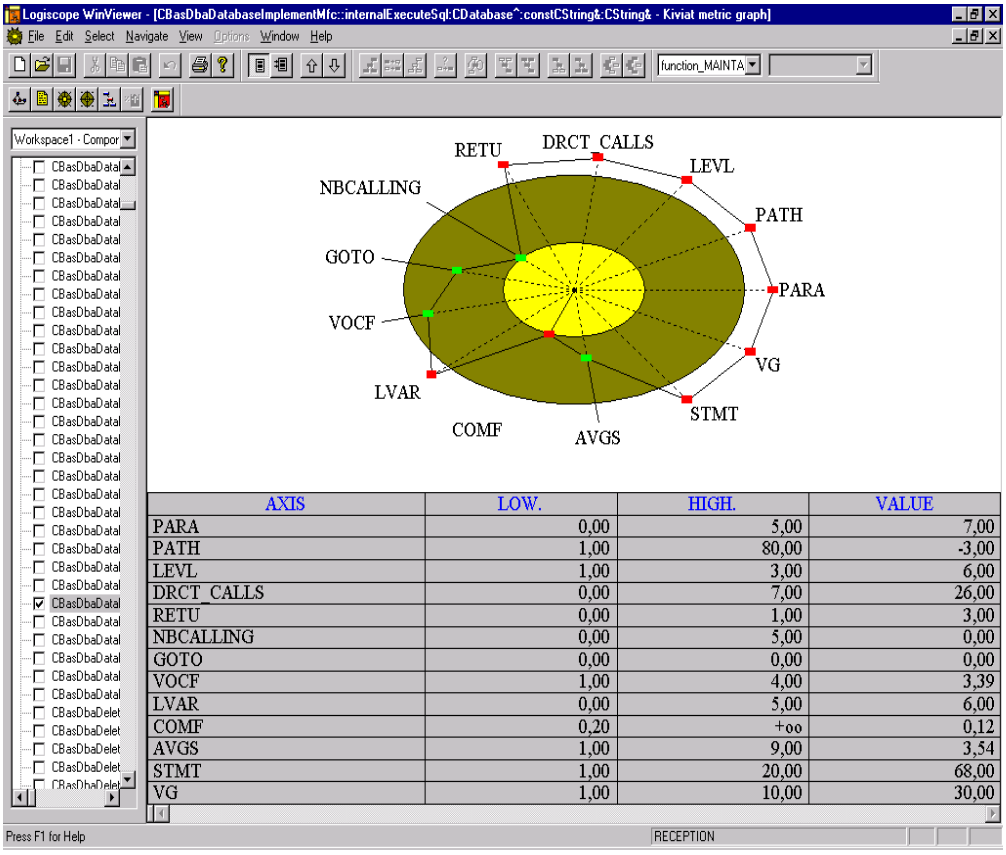
Critical values in relation to complexity and coupling between individual classes occurred in only 1.5 % of the methods. These methods include complex functionality and are part of the standard class library, and thus they have already been tested relatively often.
The aim of the static analysis is to detect critical components of a system. These are to be checked more closely during the test. Classes which exhibit a high degree of coupling and complexity are not only candidates for possible errors, they are also extremely difficult to test, as the classes possess a large number of independent paths. Testing thus becomes very expensive.
Table 6 shows the test effort.

Activity	**Effort for system tests**		**Effort for class tests**	
	Person days	**%**	**Person days**	**%**
Test case determination	13	**34.3**	8	**40.0**
Test data determination	17	**44.7**	10	**50.0**
Test execution	8	**21.0**	2	**10**
Total in person days:	**38**	**100**	**20**	**100**

Table 6. Effort

Static analysis using Logiscope is a very good way of verifying the software. The values (minimum and maximum) specified for a metrics in the quality model provide a solid basis for evaluating the software. With the aid of Kiviat diagrams, many metrics can also provide a clear picture of a method, classes or entire applications. See Figure 6.

In addition to the Kiviat metrics diagrams, it is also possible to draw up Kiviat quality diagrams. This evaluation provides important information not only for testing, but also for further implementation in the case of an increment.



Logiscope WinViewer - [CBasDbaDatabaseImplementMfc::internalExecuteSql:CDatabase^:constCString&:CString& - Kiviat metric graph]

AXIS	LOW.	HIGH.	VALUE
PARA	0,00	5,00	7,00
PATH	1,00	80,00	-3,00
LEVL	1,00	3,00	6,00
DRCT_CALLS	0,00	7,00	26,00
RETU	0,00	1,00	3,00
NBCALLING	0,00	5,00	0,00
GOTO	0,00	0,00	0,00
VOCF	1,00	4,00	3,39
LVAR	0,00	5,00	6,00
COMF	0,20	+oo	0,12
AVGS	1,00	9,00	3,54
STMT	1,00	20,00	68,00
VG	1,00	10,00	30,00

Fig. MB. 6. Kiviat diagram

Evaluation

It is premature to make any statements about the increase in **quality and productivity**. This is due to the increased complexity of the construction phase which has resulted in the generation of insufficient data for a meaningful analysis.

Positive Effects

The evaluation of the completed test cycle is positive, since testing has become a comprehensive and stable process. Testing in the early stages of the development process delivers good synergy effects between test effort and high quality of the test object.

- It is easier to find errors in the early stage of software development, because the system is not yet so complex.
- Errors which can be attributed to incorrect design can be corrected with relatively little effort.
- Excessively complex methods can be redesigned in good time, without placing undue pressure on effort and deadlines.
- The complexity of a system grows with its quality.

These effects are confirmed in Figures 5. Testing of the three prototypes also allowed the advantages of the regression tests to be exploited.

Another positive effect of the test came from testing with Cantata++. This tool allowed an error in Microsoft Visual C++ - Dev.Studio to be detected. A newly developed method, **getCurrentTime(),** was replaced during execution, and for no apparent reason, by a Microsoft method.

A further very important aspect is the fact that the test process allows <u>**the functionality**</u> and <u>**quality of a system**</u> to be consistently demonstrated in a way that has never before been possible.

The introduction of the tool Logiscope means that quality and quality features are now an objectively measurable value and no longer a collection of subjective adjectives.

Another strength of the tool is the flexible manipulation of a quality model. The quality model in accordance with Standard ISO 9126 [8] which was used can be adapted in any desired way. This allows different quality models to be developed for different systems (depending on emphasis).

Another positive aspect is the introduction of the model test and entity relationships model test. These tests, which are carried out with the support of checklists and rose scripts, allowed **72** inconsistencies/errors in design models to be identified. This allowed false associations, incorrect package assignment, missing return values in methods and infringements of the name conventions to be recognised.

Negative Effects

The negative effects of the cycle include:

- Difficulties in installing the test tools. All three tool manufacturers use FlexLm as license management. Unfortunately, every tool assumes that it has the license server all to itself. Problems therefore occurred when it was necessary to install additional license managers.
- Unfortunately the test tools too have their errors:
    - Cantata++ Ver. 2.0 cannot work with templates (STL libraries), this leads to an error.
    - In Logiscope Ver.4.0, static analysis is halted with an error when a project has been compiled using the option /YX.

These difficulties caused unnecessary extra effort and delayed our test process in the build-up phase.

Input for Test Cycle II

Cycle I supplies a range of basic metrics for our quality model which is used in Cycle II. Here the main aim is to empirically establish where for us the permissible value range lies, for example in respect of:

- Complexity,
- Coupling,
- Number instructions in a method,
- Number of derived classes,
- Number of dependent methods

etc. The measured value can also only be evaluated via establishment of the basic values.

Key Lessons learned

Technological point of view

From the technical point of view the key lessons learnt are as follows:

- Time and work spent in the tool evaluation phase is very often underestimated. Increasing competition among tool manufacturers means that many tools vary only slightly in their functionalities. Only detailed examination and analysis will reveal the differences that are very important for the process integration, e.g. internal administration of data.
- The effective way to reduce risk is to start testing early in the development cycle and to test iteratively, with every build.
- Experience with the introduction of new procedures, technologies and organisation.
- Testing and applying of the GQM method.
- Experience with the introduction of metrics.
- Experience in relation to process evaluation and tool evaluation methods
- A range of technological experience in relation to various test methods and applied tools.

The effectiveness of the test processes can be strongly influenced by two factors:

- Regression tests, which with every iteration of the development process can call upon tests already developed
- Test coverage, which should increase with every new test case

Test coverage makes it possible to check not only the progress of the testing, but also the completeness of the specification against which tests are carried out. In practice, a few UseCases need to be further specified in order to achieve complete test coverage.

Business point of view

It has become clear that an increase in productivity and quality of the software development process and the software maintenance process can only be achieved through a continuous improvement of the process rather than a single technical breakthrough.

Introducing new methods in development is a cost- and time-intensive task in the early phase of such an improvement project. Increasing effectiveness will be seen in a later phase of the Baseline Project.

With respect to quality targets, we have observed a **34 %** increase in quality over previous systems, i.e. the average error rate per 1000 statements has fallen from **6.4** to **4.2**. The tendency is increasing.

There are two main causes of this positive tendency: **model tests** and **regression tests**. These were used during the development increments and have allowed us to achieve continuous product improvement.

With respect to productivity, it is still too early to give a judgement, as the first cycle phase involved too many build-up activities. In order to determine productivity we use standard metrics, such as for example:

$$\text{Productivity} = \frac{\text{volume (product complexity)}}{\text{effort}}$$

and also additional metrics, including:

$$\text{TPM} = \frac{\text{sum (in-ouput mask)}}{\text{sum (test effort)}} \qquad \text{TPGE} = \frac{\text{sum (in-output-graphical element)}}{\text{sum (test effort)}}$$

TPM – Test productivity in relation to input-output-masks

TPGE – Test productivity in relation to input-output-graphical-elements

Various metrics derived from these were then used to measure test coverage. It was ascertained that 70 - 80 % test coverage can be achieved easily and is associated with 20 % of the test effort. This confirms the Pareto rule.

With respect to the spread of errors over the modules, we were not able to establish the Pareto rule: the errors found are distributed over **2.25 %** of the modules.

We are confident that we will achieve the target in terms of quality and productivity by the end of the project.

Knowledge of modern software technologies will increase the quality of the people involved. This will strengthen the competitive situation of a company.

Strengths and weaknesses of the experiment

The strength of the experiment is clearly the opportunity to realise the improvement process in its depths, i.e. to be able to consistently analyse, specify, translate and measure all aspects of the relevant process. Software improvement in this shape and format is unfortunately not possible in day-to-day business.

An experiment like this is the best way to evaluate new techniques and introduce them in parallel to on-going development. Even in the future we will formulate fundamental improvement actions as a separate project. Otherwise it is not guaranteed that the necessary tasks will be performed in a serious way.

The measures presented have led to a substantial improvement in the quality of the products and processes. This may be attributed to the fact that the test process was used in a systematic fashion for all part-products of the software development process, from analysis, via design, to implementation.

Conclusions and Future Actions

The new way of looking at the test and change request process with its analytical approach, delivering the values of complexity and cohesion of the tested software and process metrics in correlation to traced errors, generates a new understanding of product and process quality.

As a main aspect, the emphasis of the next actions is the definition and improvement of the change request process, implementation of improvement products (e.g. guidelines), and finally result evaluation. Internal and external dissemination have been and will be planned in the next phase. After a successful analysis of the process it will be introduced to the company as a whole and integrated into the quality management. Further long-term activities may be expected after the completion of the project.

References

[1] Balzert H., *Lehrbuch der Software-Technik,* Spektrum Akademischer Verlag, Heidelberg, Berlin, 1998

[2] Blaszczyk M., *Prozeßdefinition Ver.1.0,* DAKOSY GmbH, 1998

[3] H.Debler, *Self-Assesment Report*, DTK GmbH, 1998

[4] *Guide to Software verification and validation,* ESQPSS-0510 1/94

[5] *IEEE Standard for Software Test Documentation 829,* Publ. by IEEE

[6] *IEEE Standard for Software Productivity Metrics,* Publ. by IEEE Std.1045-1992

[7] *IEEE Standard for Software Quality Metrics Methodology,* Publ. by IEEE Std.1061-1992

[8] *ISO/IEC International Standard 9126 "Information technology – Software product evaluation – Quality characteristics and guidelines for their use"* International Standard Organisation.

[9] Morscherl I., *GQM-Plan-Software-Architekturen,* 1997

[10] Myers G., *Methodisches Testen von Programmen,* R.Oldenbourg, 1991

[11] Nordhoff S., Blaszczyk M., *Process Improvement Report Ver. 1.1,* DTK DAKOSY, 1998

[12] Ovum Report, *Software Testing Tools,* OVUM 1998

[13] Simmons D., *Software Measurement,* Prentice Hall PTR, 1998

[14] Thaller G*., Der individuelle Software-Prozeß,* bhv Verlag, 1997

[15] Wallmüller E., *Softwarequalitätssicherung in der Praxis,* Hanser, 1990

Annexes

Annex A: DAKOSY business and products

DAKOSY was founded in 1982 to support the Hamburg seaport transport sector using EDI. Currently, some 480 companies and institutions are using the system to exchange transport data with each other. The DAKOSY computer centre deals with some 400,000 computer-computer linkages each month, with a transmission volume of over 28 million data records. The IBM AS/400 systems used as transfer mainframe communicate with over 120 autonomous EDP systems from all commercial hardware manufacturers. Any computer configuration can be connected to DAKOSY using tried and tested DAKOSY standards or EDIFACT messages.

To assist the widest possible range of potential users to take advantage of "electronic commerce", DAKOSY has developed a series of advanced application systems for specific sectors. Within the transport sector, the following applications have become a byword for the professional combination of the latest EDP technology and the specific requirements of the companies working in this sector:

- ACTION	- Agents´ Container Transport Improving and Organizing Network.
- SEEDOS	- Seaport Documentation System for Seaport Forwarders.
- HABIS	- Port Railway Operating and Information System.
- GEGIS	- Dangerous Goods Information System.
- LADOS	- Liner Agents´ Documentation System.
- ZAPP	- Customs Export Monitoring in the Paperless Port.

Annex B: Author

Mr. **Marek Blaszczyk** is a project leader in DP technology at **DAKOSY GmbH**.
He was born in 1957 and graduated in telecommunications science from the polytechnic in Bydgoszcz, Poland in 1982. After college he worked as a developer of modules for electronic call systems at the "TELFA" electronics works in Bydgoszcz, Poland. From 1987 to 1996 he worked as a project leader at GEET GmbH in Hamburg, Germany. Here he gained experience in system programming, real-time systems, data transmission by radio and control of means of transport. In 1996 he joined DAKOSY GmbH in Hamburg, Germany as a project leader in DP technology. His areas of responsibility are software development, monitoring the IT market, and development of concepts for new technologies which support the strategic objectives of the company. His experience covers many areas, including quality assurance, process improvement and management, systems analysis, system design, implementation and testing.

TERRA FIRMA

Improvement of testing process through systematisation for increasing software manufacturing assurance

Julián Navas
Alberto González

Sainsel, Sistemas Navales, S.A.
Manuel Velasco Pando, 7
41007 - Seville - Spain
www.sainsel.es
terra_firma@sainsel.es

Introduction

Nowadays, Testing Techniques for Software products are essential to improving the reliability of applications. Sainsel, as a manufacturer of complex systems for Naval Command & Control, Simulation and satellite-based Fleet Monitoring, is aware that the early detection of errors in the production process is vital to reducing cost, shortening delivery times and decreasing the number of corrective after-sale interventions.

The purpose of the experiment is to implement an integrated test environment and to determine in quantitative terms (using different metrics) the impact on economic aspects and productivity. The experiment also aims to bring about a substantial improvement in the reliability and quality of the applications delivered to the end user. Different metrics are used: Project Cost per Line of Code (LOC), Application Maintenance Cost, Testing Proficiency Ratio, etc.

The implementation of the test environment includes various phases of the System Software Life Cycle (SLC): Specification and Analysis of functional requirements, Design and Programming, Verification & Validation and Operation & Maintenance. Different tasks are carried out: implementation of the process to analyse and correct specifications, test case specification and design (Black Box), White Box tests, acceptance tests, test management, etc.

At the end of the project we will have learnt:

- A practical method for writing testable requirement specifications.
- How to specify and design a set of cases to verify that the design and code fully implement all functional requirements.
- How to use powerful tools, such as SoftTest and LDRA.

We expect a considerable reduction in errors in the end product and a significant decrease in maintenance costs.

We expect to consolidate an independent software testing team.

MTP, Métodos y Tecnologías de Sistemas y Procesos, S.L. (www.mtp.es), a Spanish consulting firm which is highly experienced in quality issues, is acting as a subcontractor, providing Sainsel with the training needed in Requirements Specification, Test Case Design, and Black Box Test Case Execution, as well as helping to define and exploit the standard set of metrics. MTP provides a variety of training and other assistance services in support of those organisations involved in Software Process Improvement and is very experienced in the introduction of metrics models.

TERRA FIRMA as such has been supported by the European Commission within the framework of the ESSI initiative, and has been advised by the Spanish SPINODE in the preparation of external dissemination.

Business Motivation

From the business point of view, Testing Techniques for Software products have been essential to improve the reliability of applications and relating TERRA FIRMA Project objectives to company objectives.

It is essential–for reasons of cost–that the number of latent errors which appear in the product in the operation & maintenance phase be as small as possible, because the correction of errors and the reinstallation of software is very expensive. The correction of defects or errors is even more costly if it has to be carried out at the client’s facilities, which may be geographically distant, i.e. in African or American countries.

Sainsel has a strong presence in the Spanish Market, but has increased its international market share in recent years, offering its products mainly to the African and South American markets. Obviously, Sainsel faces strong competition, not only from outside Europe but also from European companies. This means that investments must be increased, developing new leading products, but taking into account that these emerging products must be competitive and of a high quality.

With a view to achieving this goal, Sainsel is seeking to reduce maintenance costs and increase the reliability of systems/products, taking into account that safety is one of the most important requirements requested by its clients. By achieving these objectives, Sainsel could increase the loyalty of its clients and maintain their trust over the years.

Objectives

The aim is to bring about a significant improvement in the quality (reliability) of applications using an integral test plan to substantially improve test coverage (both in functional and structural terms). The increase in reliability will be achieved for reusable components as well as for end products.

Methodologies and Techniques:

- Draw up unambiguous system requirements.
- Design the functional and acceptance test cases (Black Box Techniques)
- Defect tracking using regression library.
- Design code coverage tests and analyse code (White Box Techniques)
- Verification & Validation (Traceability).

Another objective is to analyse Cost/Benefit, by measuring cost and productivity parameters in order to quantify the cost–of the quality–of the improvements introduced.

Metrics used to measure impact on the final goals [2]:

- Financial impact: present cost; increment of the cost due to implementation (training, tools, new resources, etc); cost of the application in the maintenance phase.
- Impact on productivity: number of LOC or Function Points (FP) delivered per unit of effort (LOC/man.days); effort needed to support a LOC or FP application (man.days/LOC); Duration Delivery Rate, which measures how quickly LOC or FPs are delivered (LOC/Elapsed Time); application maintenance load per person, which measures the overall work load of maintenance and support personnel (LOC/number of maintenance people).
- Impact on quality: cost to repair defects; stability ratio, which measures how well the application met the expectations of the user (number of changes/LOC); testing proficiency ratio, which measures the efficiency of the testing technique and also system defects as a percentage of total FPs or LOC.

Starting Scenario

TERRA FIRMA is focused on improving the software process. This experiment was prompted by high maintenance costs and the lack of a dedicated Test Department.

The Sainsel Software Department is carrying out this experiment, which involves four software engineers skilled in programming and design working as part of the testing team.

Currently, the software process in Sainsel includes five points:

- Project Management: Ms-Project and CVS are used in some projects, but not all.
- Methodology: Object Oriented Methodologies are used in the design phase of SLC and are supported by Rational Rose.
- Code: Visual Basic, Ada, C and C++ programming languages are mainly used for coding. Unix and NT Operating Systems are usually used.
- Tests: unit, integration and acceptance tests are implemented in all projects.
- Metrics: quality and productivity metrics are not collected in a systematic way. Metrics cost, effort and LOC are available.

The experiment is being carried out in conjunction with a real system project (CSP98 project): “Fishing Fleet Tracking and Monitoring System”. The baseline project is characterised by world-wide coverage and twenty-four hour uninterrupted operation. By means of a device installed on board a fishing vessel, with satellite tracking technology, from a Control Centre it is possible to monitor: geographical position, information about the state of the fleet, speed and course, the activity that the fishing vessel is performing at that moment, entries to and departures from port and fishing areas, etc. Therefore, it is very important that this system is highly reliable. The required reliability can be achieved through improvements in the test process [3].

Due to the fact that the CSP98 project involves two builds, the TERRA FIRMA project is divided into two parts, with each phase of the experiment being carried out twice.

CSP-98 is a medium-size project in Sainsel: ~100 KLOCS of executable code, 12 months duration, and 2000 person/days of effort.
Tools and languages used:

- C++ and Visual Basic languages.
- Solaris 2.6 and Windows NT.
- Informix.

Description of the experiment

Overview

TERRA FIRMA is divided into two main phases [3]:

Phase 1, Implementation of the Test Process, covers from the specifications phase through to delivery of the product (CSP-98). It is necessary to define the activities, methods and tools used in each one of the SLC phases [5,8].

The purpose of Phase 2, Management of Implementation, is to control and measure in quantitative terms the effectiveness of the implementation of test systematisation. As commented previously, the economic, productivity and quality rates will be calculated and monitored to determine the impact of the implementation of a test environment on the above-mentioned rates as well as on cost / benefit.

Phases of the experiment

The activities included in the two phases of the TERRA FIRMA Project are explained below:

- Specification and analysis of functional requirements.

  At the same time as the User Requirements are drawn up, the graphics of the method cause/effect are constructed [10]. Any language pitfalls, negations scope, omissions, ambiguous logical operators, etc. will be corrected and reviewed.

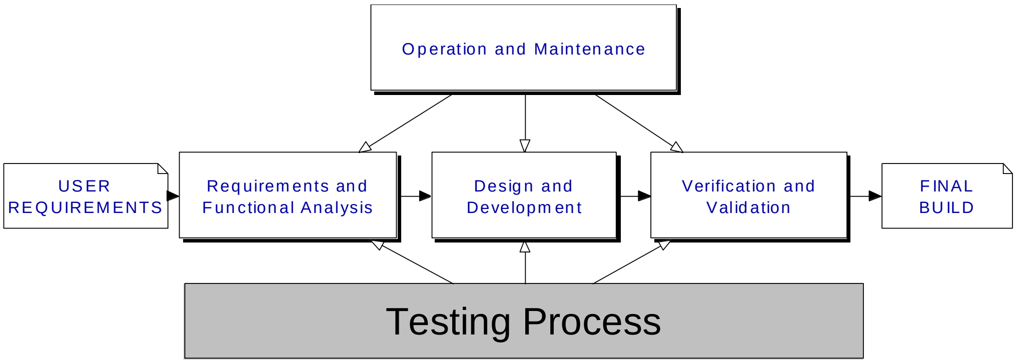

Operation and Maintenance
USER REQUIREMENTS
Requirements and Functional Analysis
Design and Development
Verification and Validation
FINAL BUILD
Testing Process

Fig. TFIRMA1: Test Process

A repository of requirements has been implemented using Lotus Notes to record the derived text information of the graphics constructed on the basis of the functional requirements [4,5] and to control the different versions generated as a result of changes in the different phases of the CSP98 project.

The next step is to specify and design the Test Cases using Black Box Techniques (Cause / Effect Method, Equivalence Class Partition, Bound Values Analysis and Defects Inference) [10]. The functional requirements will be covered 100% (it is done in this way because it is based on mathematical algorithms, which use logical operators as in hardware tests that are known for high reliability). A powerful tool (SoftTest) has been used to design different functional variations that define all test cases.

In the same way, a repository has been created using Lotus Notes to record and control test cases. This repository will be the source of test executions and defect control.

- Design and programming.

During the design and programming phase, the Software Testing Team builds the test cases specified and designed in the previous phase, preparing sets of input test data in an independent "target" system (analogue to the client’s system). Data is normally entered in a "target" database or using an ODBC link ("target" database - Ms Access).

Once the test cases have been performed, the Software Testing Team records any errors and defects in the test case database, so that they can be later analysed to obtain quality metrics.

White Box Tests [6,7,9] are performed to complete Black Box Testing using structure analysis techniques and measuring code, statement, decision and LCSAJ coverage. If coverage does not reach an established minimum, more White Box Tests are designed using an LDRA Testbed.

Finally, using an LDRA Testbed, design faults (data and control flows, statements, syntax, etc.) can be analysed, studying Code Complex (Cyclomatic Complex (McCabe), Knots, Nest, LCSAJ) and Sainsel C++ programming yardsticks and requirements. For example, if the cyclomatic complex of a block is high (more than ten), that block must be split into simpler blocks.

- Verification & validation.

At this stage, the Software Testing Team has verified that the system is working according to the functional specifications and has reviewed all the paperwork. This is done by testing all the functions of the system ensuring traceability between functional requirements and test cases and that test case results are as expected, etc.

- Operation & maintenance.

  The maintenance phase covers the entire process. Databases might be modified due to changes in requirements or to results showing defects found during performance of the tests. These modifications make it necessary to repeat the previous processes depending on the point at which they occurred.

The implementation of any improvement process in a company must have at least the following two goals: first, to optimise implementation and maintenance costs; and, second, to improve productivity and technical aspects. With these two goals in mind, those involved in the project analyse impact at three levels: economic, productivity and quality.

The results obtained by implementation of the process are evaluated according to several metrics:

- **Productivity Metrics**

Project Delivery Rate measures the LOC delivered (KLOC/person · year)	$\frac{LOC\ Delivered}{WorkEffort}$
Application Support Rate measures the effort required to develop support applications (man year/Application LOC) where Application LOC are the LOC used to support the CSP-98 project	$\frac{WorkEffort}{Application\ LOC}$
Duration Delivery Rate measures the speed of code delivery (LOC/Elapsed time)	$\frac{LOC}{Elapsed\ time}$
Application Maintenance Load per Person measures the overall work load of support personnel (KLOC/man)	$\frac{LOC}{Maintenance\ people}$

- **Economic Metrics**

Project Cost per LOC measures the cost incurred by development (Euros/KLOC)	$\frac{(WorkEffort \cdot cost) + Other\ \ expenses}{LOC}$
Application Maintenance Cost measures the cost of application maintenance and support (Euros/KLOC)	$\frac{(WorkEffort \cdot cost) + Other\ \ expenses}{LOC}$

- **Quality metrics**

Repair Cost Ratio measures the cost required to investigate and repair defects (Euros/KLOC)	$\frac{Repair\ time \cdot Cost}{LOC}$
Stability Ratio measures how the well the application meets the expectations of the user (no dimension)	$1-\left(\frac{Defects\ Reported}{FP}\right)$
Defect Ratio measures the quality of development and application enhancements delivered to user (Defects/KLOC)	$\frac{Number\ of\ defects}{LOC}$
Testing Proficiency Ratio measures system defects as a percentage of total LOC (Defects/KLOC · Tests)	$\frac{Defects\ found\ by\ user\ testing}{LOC}$
Cyclomatic Complexity is a measure of the complexity of the directed graph corresponding to each function [9].	$edges - nodes + 2$
Average size per procedure	$\frac{LOC}{number\ of\ procedures}$
Comment density measures the comment LOC – Project LOC ratio [11].	$\frac{Comment\ LOC}{LOC}$

LOC: lines of project executable code.
Application LOC: lines of support application executable code.
Work Effort: Working days / Working months.
Elapsed Time: time required to develop System / Software.
Cost: person cost.
Tests: acceptance tests designed.
FP: Function Point = 128 LOC (C language) [11].
Edges: edges of directed graph [7]
Nodes: nodes of directed graph. [7]
Defects Reported: functional defects reported by user (over six months)

Current state of the experiment

All the phases of the TERRA FIRMA project must be carried out for the first build and the final build. At present, the base project has reached the maintenance and operation phase for the first build and the requirements specification phase for the final build.

In the TERRA FIRMA project, all the test process implementation phases (systematisation) have been completed for the first build of the base project and the test process implementation management phase has been reached. Quality and productivity metrics have been obtained using the CSP98 project results for the first build.

During implementation of the test processes the following tasks were completed:

- Draw up system requirements.
- Design the functional and approval test cases.

- Test scripts automation (SoftTest from Bender & Associate).
- Implement the requirement coverage matrix.
- Create a testing and requirements repository using Lotus Notes.
- Build the test cases.
- Execute the functional test cases.
- Design the white test cases.
- Analyse code coverage using Testbed LDRA
- Build the test regression library.
- Create a defects and errors repository using Lotus Notes.
- Results verification and validation.
- Execute the approval test cases.
- Finally, integrate all the databases in one (testing, requirements, defects and errors databases).

During test process implementation management the following tasks were completed for the first build:

- Write documents and papers based on MIL-Std-498 (Requirements Specification Document, Test case Specification Document, Approval Test Document, Result Test Document, Validation &Verification Report, Test Case Maintenance Report).
- Obtain quality and productivity metrics.
- Provide training in different techniques: "Writing Testable Requirements", "Black Box Testing" and "White Box Testing".
- Evaluate different tools: LDRA, SoftTest, TestDirector, Caliber, Panorama, Cantata, etc.

Metrics Results

These metrics have been calculated for the first CSP-98 project build using currently available data (LOC, effort, defects found, development time, etc.). These values will be compared with average values generated by projects carried out at Sainsel previously.

The metrics relating to project costs or maintenance cannot be calculated until the CSP-98 project has been completed.

Project Delivery Rate	
$1.28 \ \frac{KLOC}{Man \cdot Month}$	$58.08 \ \frac{LOC}{Man \cdot Day}$
Duration Delivery Rate	**Application Support Rate**
$106 \ \frac{KLOC}{Year}$	$0.23 \ \frac{Man \cdot Month}{KLOC}$
Defect Ratio	**Testing Proficiency Ratio**
$6.34 \ \frac{Defects}{KLOC}$	$0.15 \ \frac{Defects}{KLOC \cdot Test}$

Stability Ratio	**Repair Cost Ratio**
94%	$608.26\ \frac{Euros}{KLOC}$
Average Cyclomatic Complexity	**Average size per procedure**
2.22	*30.36 LOC*
Comment density	
26%	

- System development rate: average value: -> 1.4 - 61.65

- Effort coefficient for support tools: average value -> 0.27

These three metrics indicate that the effort expended to develop code in the CSP-98 project is more or less average, signalling that it is representative of projects carried out at Sainsel in terms of productivity.

- Duration delivery rate : average value -> 59.45

An increase of approximately 80% on the average value is achieved. This is due to the number of programmers working on the project. The use of visual tools that generate large amounts of automatic code is another factor that influences this result.

- Repair cost ratio: no previous data available.

- Defect ratio: average value -<4.08.

The number of defects found during the test process exceeded the average value by 50%. This is a good result and proof of how exhaustive the process is.

- Acceptance test efficiency ratio: average value ->0.17

This is a slight improvement on the average value. This metric depends largely on the number of tests performed during acceptance testing. In this case, a comprehensive System Acceptance Testing Document was produced, which met the client’s requirements amply. The result obtained, although better than the average value registered by the company, did not meet the expectations of the testing team, since some of the system functionalities were not implemented.

- Stability ratio: average value ->88%

The result is above average, although only the modifications received over the last three months have been taken into account.

- Stability ratio: average value ->2.07.

The result is acceptable since is it well under 10, which is the maximum recommended value per procedure.

- Average size per procedure: average value ->40.9

There is a reduction in average size per procedure. The code for each function is less dense and more easily understandable, and therefore less likely to give rise to errors. The maximum recommended value per procedure at Sainsel is 100 LOC.

- Comment density: average value ->26%.

There was a considerable increase in comment density owing to the headers introduced by the CVS version control tool, which translates into a more intelligible code. The recommended proportion of comments is between 20 and 30%.

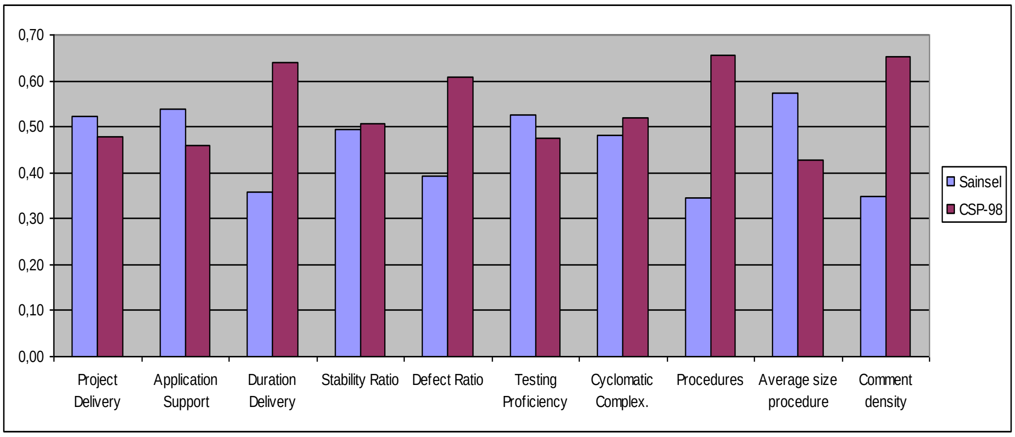


Fig. TFIRMA2 Comparative metrics

Statistics

- Accumulated test results by version.

	Version 0.95	**Version 0.96**	**Version 0.97**	**Version 0.98**
Satisfactory Testing:	30	31	31	93
Unsatisfactory Testing:	42	85	88	137
Executed Test Cases:	**72**	**116**	**119**	**230**

Satisfactory Testing = Test cases with satisfactory results.
Unsatisfactory Testing = Test cases with unsatisfactory results.
Executed Test Cases = All the executed test cases.
Defects = Errors + Improvements.

- Evolution of defects according to gravity.

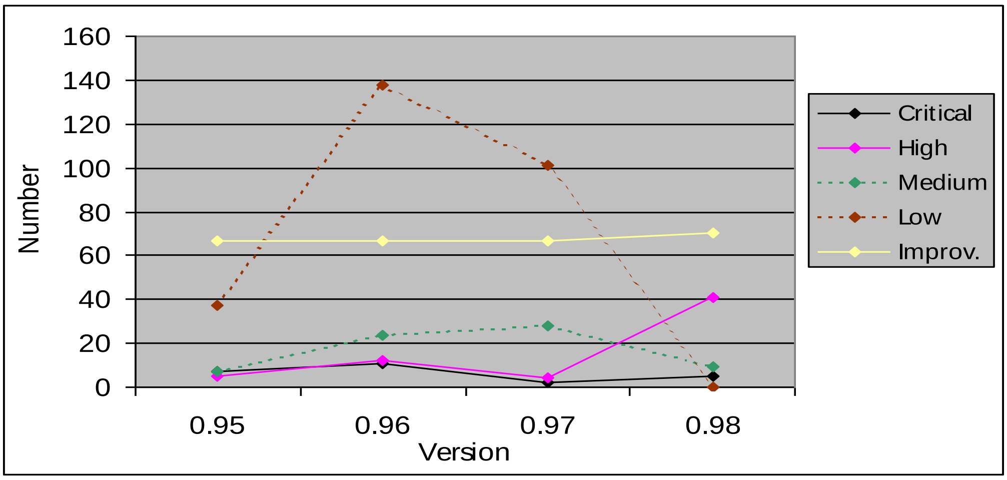


Fig. TFIRMA3 Defects per version.

It can be seen in figure TFIRMA3 that open defects exist in version 0.98. This is due to the fact that certain functionalities were not implemented in this build. Therefore, while critical errors are seen to increase initially due to the implementation of the main system functionalities, they tend to decrease for later versions. However, the evolution of high-gravity errors is quite different, with the development of a number of new functionalities in version 98 giving rise to a considerable increase in such errors. Medium- and low-gravity errors evolve in an acceptable manner according to expectations. The incorporation of improvements is reserved for the final build, so that the number remains practically unchanged, with only a slight increase due to the implementation of new functionalities.

- Evolution of Defects Vs Test Cases

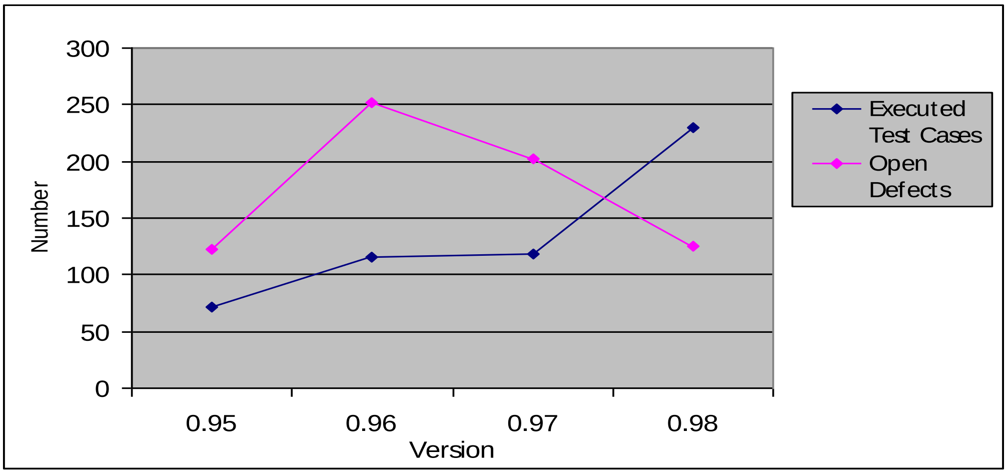


Fig. TFIRMA4: Defects Vs Executed Test Cases per version.

Figure TFIRMA4shows both the number of test cases performed and the number of open defects (errors and improvements) in relation to the software version developed.

Conclusion and future action

The experiment bridges a real gap in our organisation: the need to consolidate a Software Testing Team.

Nevertheless, there are some factors that have not assisted in the development of the TERRA FIRMA project because of a lack of previous experience. The introduction of the TERRA FIRMA project meant that a number of personnel aspects had to be addressed: people must be made aware of the fact that what is being measured is the baseline project, and not the people involved. In order to obtain the full collaboration of the whole team, it was necessary to explain how the experiment fits in with the organisation's medium-term and long-term goals, the benefits of introducing the new methodology, and how the return on investment is expected to be obtained.

It was also necessary to clarify that the experiment will not affect the course of the CSP98 project and that the necessary resources have been assigned to carry out additional work. It is therefore evident that the notion of SW testing is not an easy one to grasp.

Since the Base Project (CSP98) is standard in terms of time, documentation, etc, all the techniques and procedures can easily be implemented in any other similar projects.

In June the first build of the CSP98 project was installed, and good results have been registered in the maintenance phase to date. However, it is still too early to draw any firm conclusions about system maintenance and the stability ratio.

The metrics results are good overall. An improvement on these results is expected when post-delivery maintenance costs are taken into account. These will be calculated once the second build is completed.

The Software Testing Team is currently disseminating its experience inside and outside Sainsel. Sainsel plans to incorporate the procedures derived from the TERRA FIRMA project in its QP (Quality Procedures), in order to promote their use by everybody throughout the company and to develop a strong Software Testing Team and a sound testing methodology.

References

[1] http://www.esi.es/VASIE/

[2] Fenton N. Software Metrics (A Rigorous Approach), Editorial Chapman&Hall. ISBN 0442313551

[3] TERRA FIRMA – 27703. Testing process through systematisation for increasing SW manufacturing assurance. Annex I.

[4] Guía para la Elaboración de un Documento de Especificación de Requisitos SL/04/0101/PSW/003/01.

[5] MIL-Std-498.

[6] Computer Science and Technology Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric NBS Special Publication 500-99

[7] Arthur H. Watson&Thomas J. McCabe. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric. National Institute of Standards and Technology Gaithersburg, MD 20899-0001 September 1996. NIST Special Publication 500-235.

[8] Pressman, R, Software Engineering: A Practitioner's Approach, McGraw Hill,1992.

[9] LDRA Testbed v 5.6.0. Reference Manual.

[10] SoftTest Release 5.3 Reference Manual.
[11] COCOMO II Model Definition Manual.