EuroSPI Conference DAY 1

Nordic Hall 1	Nordic Hall 2

08.30 - 08.45 **Welcome**
08.45 -09.15
Hakan Wickberg, **Head of SPI, IVF,** *The Status of SPI in Swedisch Industry and Scandinavian Achievements*
09.15 -10.45 **Key Note Session 1**
Tilo Messer**, Siemens,** *Software Process at the Gate to the Top*
*Paul Rogoway***, Motorola,** *Adding SPICE While Preserving CMM*

10.45 - 11.15 **Coffee break** (in exhibition area)

Nordic Hall 1	Nordic Hall 2
11.15 - 12.45 Session 1 **(S1) SPI & Systems Development Part I** Dejan Zivkovic**, Alcatel,** Belgium *Introducing Risk Management* Joern Muenzel, **Robert Bosch GmbH**, Germany *Standardised Test Programming in SystemTest* Tero Lindholm, **Nokia**, Finland *Combining Business Process Improvement And Systems Development*	11.15 - 12.45 Session 2 **Metrics Driven SPI Part I** Otto Vinter, **Bruel & Kjaer**, Denmark *Defect Analysis to Initiate SPI* Terttu Orci, **DSV,** Sweden *Software Metrics Applications in a European Perspective*
12.45 - 14.00 **Business Lunch**	
14.00 - 15.30 Session 3 **Implementation of SPI Part I** Tor Stalhane, **SINTEF,** Norway *Data Driven Improvement for SME's* Bjarne Mansson**, Barco AS,** Denmark *Years of SPI Experience* Alessia Billi**, Sodalia,** Italy *Experience with the Installation of an SEPG*	14.00 - 15.30 Session 4 **Metrics Driven SPI Part II** Antonios Tsipianitis, **TEGEA**, Greece *Improving Civil Engineering Through Metrics* Brian Chatters, **ICL**, UK *Metrics and Risks in SW Systems Integration* Kenneth Kvinnesland**, Navia Aviation,** Norway *Application of Metrics in Safety Critical Fields*
15.30 - 16.00 **Coffee break** (in exhibition area)	
16.00 - 17.30 Session 5 **Implementation of SPI Part II** Jorn Johansen, **Delta,** Denmark**,** Lars Mathiassen, **Aalborg University**, Denmark *Lessons Learned in a Natioal SPI Effort* Bill Culleton, **3S**, Ireland *SPI By IPS - Involvement, Planning and Structure* Seija Komi Sirvio, **VTT**, Finland *Experiences from Practical SPI*	16.00 - 17.30 Session 6 **Object Oriented SPI** Paul Sullivan, **ESBI**, Ireland *Impact on Introducing OO SW Development Methodologies* Sten Jacobson, **Rational**, Sweden *The Unified Software Process* *Paolo Caricchia*, **Aeroporti di Roma**, Italy *Object Oriented System Integration*

20.30 **Social Event**

EuroSPI Conference DAY 2

Nordic Hall 1 — Nordic Hall 2

08.30 -10.45 **Key Note Session 2**

Keith Dyne, **Ericsson System Software Initiative (ESSI)**, *Benetits from Continuous Software Improvement*

Giselle Roesems**, CEC DG III,** *A Summary of ESSI (European Systems and Software Initiative) Experience in Europe*

10.45 - 11.15 **Coffee Break** (in exhibition area)

Nordic Hall 1

11.15 - 12.45 Session 7

SPI Experience from Small Teams

*

Beatrix Barafort, **Centre de Recherche**, Luxembourg
A Small Developer's Framework
Joao Battista, **ISCAA,** Portugal
CMM in a Micro Team
Svein Are Martinsen, **Invenia**, Norway
Improving Estimation and Requirements Management

Nordic Hall 2

11.15 - 12.45 Session 8

Information and Team Management Solutions for SPI

*

Richard Messnarz, **ISCN**, Ireland + Germany
SPI in network based Quality Assurance Environments
Atsuo Hazeyama, **NEC Corporation,** Japan
Promotion of an ISO9001 based quality system using the WWW
Janos Ivanios, **Memolux**, Hungary
The PASS Experiment in Hungary

12.45 - 14.00 **Business Lunch**

Nordic Hall 1

14.00 - 15.30 Session10

SPI & systems Development Part II

*

AbrahamPeled, **Motorola**, Israel
Establishement of Defect Prevention Mechanisms
Malgorzata Warne, **Ericsson**, Sweden
Practical Implementationof a Cleanbase Process
Reidar Conradi, **TELENOR**, Norway
Re-Use of SW Development Experience

Nordic Hall 2

14.00 - 15-30 Session 9

SPI and SW Life Cycle Support

*

Antti Valimaki, **Valmet**, Finland
Enhancing software configuration management for a process control system
P.J. King, K. Arthur , **Clockworks,** Ireland
Project Management and Engineering Control System
G. Roth, **Transaction**, Germany
MultiPlatform Configuration Management

15.30 - 16.00 **Coffee Break** (in exhibition area)

Nordic Hall 1

16.00 -17.30 Session 12

SPI Processes and Modeling

*

Ulrich Zanker, **Lindenberg aerospace,** Germany
Experience with Dynamic Systems Modeling in Integrated Tool Support
Christian Zwanzig, **AB Bremen**, Germany
Modeling and design Guidelines for Outsourcing Projects
Clemens Gasser, **Joanneum**, Austria
***CCM** - A fundamental Process for Improving Quality*

Nordic Hall 2

16.00 -17.30 Session 11

SPI on Personal + Hollistic Level

*

Charalampos Avratoglou, **Computer Logic SA,** Greece

Plus Open Places for Contributions from Attendees

17.30 - 18.30 **Closing Panel**

Remember to use the Strengths as SPIi Drivers

Software Process Improvement as an Important Factor in Business Success

Bo Balstrup, **Danfoss**, Denemark, Anne Mette Jonasen Haas, Jorn Johansen, **Delta,** Denemark, Richard Messnarz, **ISCN**, Ireland, Risto Nevalainen, **STTF**, Finland, Tor Stalhane, **SINTEF**, Norway, Hakan Wickberg, **IVF**, Sweden

Session 1 – Systems Development Part I

Introducing Risk Management in Alcatel SSD S12

Dejan Zivkovic, S12 SSD SEPG
Alcatel SSD, Antwerp
Risk Management Council Chairman

Improving the System Test Process

Jörn Münzel
Bosch Telecom, Frankfurt, Germany

Implementing SPI: Combining business process improvement and system development at Nokia

Tero Lindholm
Nokia, Salo Finland

Introducing Risk Management in Alcatel SSD S12

Dejan Zivkovic, S12 SSD SEPG
Alcatel SSD, Antwerp
Risk Management Council Chairman

Introduction

Alcatel SSD (Switching Systems Division) S12 organisation develops and maintains 2000K lines of code big public switching system and several surrounding products of smaller size. It runs in parallel tens of releases and projects across more than 10 development sites on 4 continents for customers all over the world. Typical project takes several tens of person-years of effort.
The organisation has planned and executed a series of CMM-based software process assessments in all its locations. A main SPI program has been put together and budgeted for the period of 3 years. The objective was to significantly increase the process maturity level of the SSD organisation, using CMM as the guidance.
Risk management has been recognised for its improvement from the very beginning of the SPI initiative. A separate activity within Project Management Working Group has been started up with the task to investigate and propose risk management (RiMa) process.
The purpose of this article is to share the experience of the introduction of this subtle process into the large organisation such as Alcatel SSD.

Piloting

Initial Process Definition

Since the organisation did not have much of experience about RiMa, the activity started by collecting published articles and books about RiMa. There were contacts

made with several consulting companies. After brief evaluation, based on the collected information, one was contracted with the request to help the organisation in developing its own risk management process.
The process has been described within 3 months after the start of the activity. It was based on simplified Charette's model [1] and included the following steps: identify,

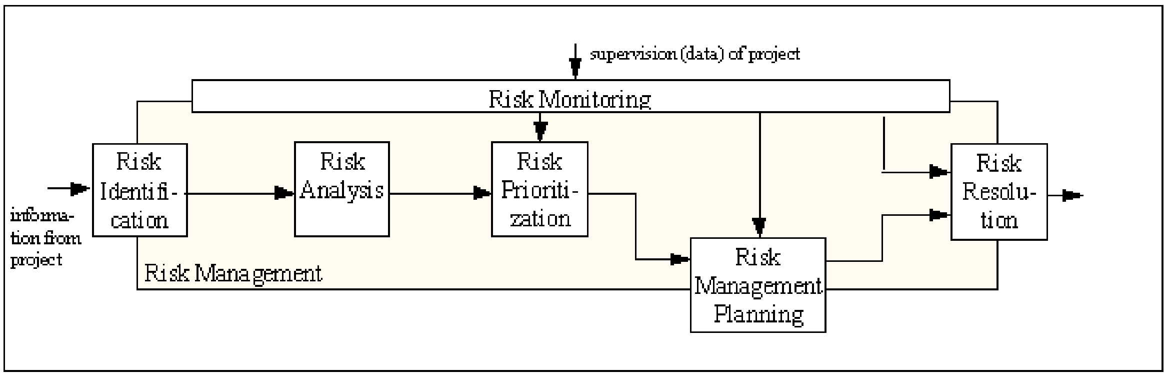


Figure ZIVKOVID.1: Risk Management Process

analyse, prioritise, plan, monitor and resolve (figure 1).
The purpose of the introduction of this specific process is to improve project management. In particular, the predictability of the project quality, milestones and cost had to be improved. In the long run, a better communication with customers, transparency of our processes and customer satisfaction must be achieved.
Two projects have been chosen for piloting starting several months one after the other. Both were run in the same location. The two differed in the priority given to them, departments and line management that was in charge.

Choice of Pilot Projects

The first of the 2, referred to as X1, was a typical example of a good pilot. It was a project for one of the organisation's main customers, critical for both customer and supplier. It got one of the first priorities, all the required management attention and it was able to make use of all the other early SPI results, such as improved requirements management, focused inspections of software work products, advanced module test, improved and automated test process, newly defined internal qualification test.
The other, X2, was X1's opposite. It was run with far less attention, was accepted from the beginning to be late and it was not consistently stimulated to use SPI results.
As both projects were approaching their hand over dates, some results could be outlined:

- X1
    - Overall project performance was good.
    - very important risks were successfully mitigated.
    - An observation has been made that X1 project leader was "born to be a risk manager".
- X2
    - Project performance was (expectedly) poor.
    - RiMa process did not perform according to the process description and quite often did not sustain its activities faced with overall project difficulties.
    - However, some risks were reasonably mitigated.

Lessons Learned

After the pilots, the local SEPG organisation was given the task to investigate RiMa performance in both projects. Particularly the question was raised "what went wrong with X2".
Not surprisingly, the investigation confirmed that what went wrong with X2 could not be attributed to the failure of RiMa process in particular, nor to any other subprocess. On the contrary, to the degree of the RiMa process application, both projects may say that they performed RiMa reasonably.
Other experiences:

- Given the environment, investment in RiMa in later development phases has little sense. Projects are still diving in day-to-day problem management and there is little time and attention for a proper RiMa. Nevertheless, some "sanity checking" may be continued:
    - Follow up and closure of open actions
    - A question "what more could we do to prevent later problems" is always reasonable.
- Common feeling is that RiMa is a good practice. However, any trial to put any hard figure on that is easily challenged and disputed.

First Institutionalisation Trial

Based on overall success of X1 and its good experiences with risk management and not a bad RiMa performance of X2, a decision has been taken soon after the evaluation of the pilots to put the infrastructure in place in order to institutionalise RiMa across corporate locations. This has taken place within the following months and RiMa Business Subsystem was structured according to the given figure 2.
On the left side of the organisation one may observe a set of PRiMaTs (PRiMaT = Project Risk Management Team). A team of (typically 2-4 people) had to be nominated to drive and execute RiMa process. The team would be typically composed as a subset of project staff, although this would be not a pre-requisite. RiMa Council, on the right side of figure 2, was at that time nominated from at least a coach per location and a coordinator at the corporate level.
The task of coaches was primarily to guide local PRiMaTs in applying RiMa process.

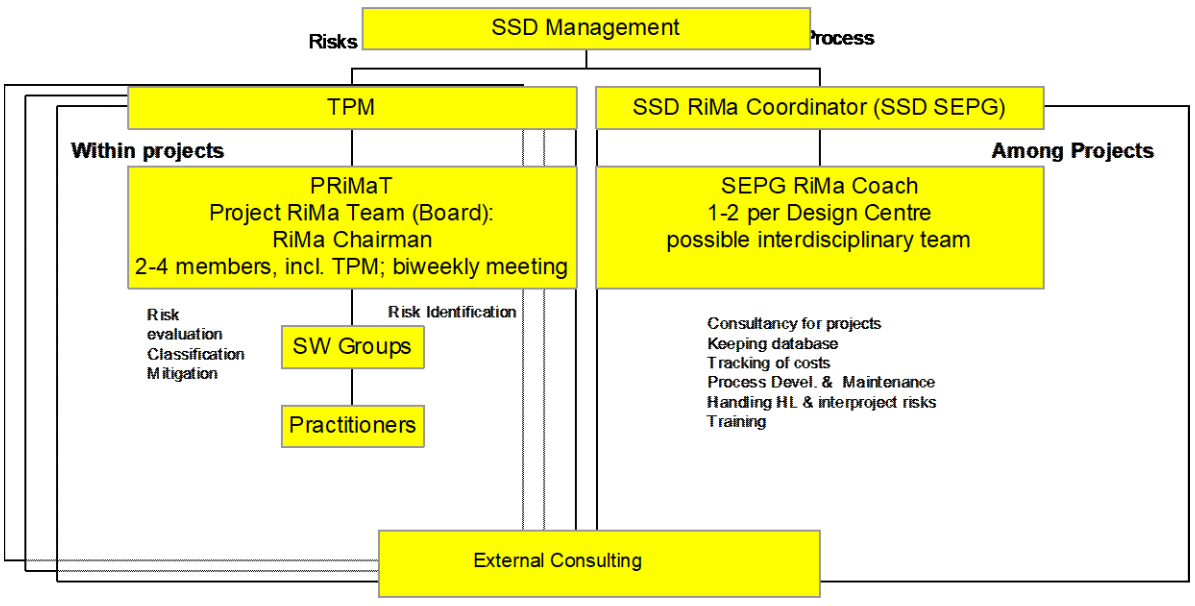


Figure ZIVKOVID.2: Risk Management Business Subsystem, with PRiMaT's and a Council

Secondly, coaches had to re-direct risks based on their scope: low-level risks were communicated to the individuals and functional groups which were expected to deal with them; high-level risks (which on their turn could have been cross-project risks and cross-location risks) had to be communicated to senior management, possibly with proposed action plan. Those which were judged to be at middle, project level, had to be treated by the PRiMaTs and project participants.
These agreements were negotiated and confirmed and a preliminary list of projects to run RiMa was made. The criterion used was “all the new projects which are not later than in high level design phase”. That meant practically that 11 new projects in 8 locations needed to “join the club”. This was to a certain extent achieved within 3 months. A 2 day training course for 16 participants was organised in January (coaches, PRiMaT leaders and members). They were expected to disseminate the process, e.g. to organise training along the kick-off meetings for PRiMaTs when a new project would be started. In the months to come more projects joined.

What Has Happened?

The most of the projects that exercised RiMa, have started doing it quite enthusiastically. The principles of RiMa were judged to be the right ones and helpful to the project. Doing it was perceived also to be maintaining the good atmosphere throughout the project.
However, the world of S12 is ever-changing world where not only many projects compete for scarce resources, but also several initiatives for process improvements are being launched in parallel. Given the nature of the RiMa process, in order to sustain it or to get the most benefit out of it, different projects have taken different tactics.

- Some projects tried to identify and mitigate as many risks as possible.
- Some projects tried to do something without formally logging what they were doing.
- The most of the projects tried to cope with the risks largely on their own (within the scope of the project organisation).
- Some were following the actions stemming from RiMa through the RiMa process, others have incorporated the follow-up in the “normal” project management and reporting.

More than a half of the projects was not able to sustain the RiMa activities throughout the project life cycle.
Common to all the projects was a high number of “short term” risks, which were in essence known-to-be problems from the past experience.
Management who sometimes was inpatient to get an answer to the question of ROI of this process was either left with no answer, or they considered the examples of calculated ROI that were provided to them as inadequate.

Examples

Project X3 started RiMa activities in February. It stopped RiMa in May during design with the following results:

- 28 risks identified and tracked
    - 1 successfully mitigated
    - 3 turned to obsolete

- 9 other triggered with no evidence of results

Project R81 started in January. It stopped in August during TLD with the following results:

- 48 risks identified and tracked
    - 1 successfully mitigated
    - 11 risks materialised as problems
    - 11 risks became obsolete
    - 14 others were triggered and some of the corresponding actions were closed; no further evidence is provided about the outcome

This kind of picture was similar in the several other projects.
A better example is a project Y1 which started in later the same year. After more than a year of successful activity, the project still continued with RiMa through the first implementation phase. The status in before hand-over:

- 17 Risks were identified and tracked
    - 14 Risks were successfully mitigated
    - 3 Risks are triggered

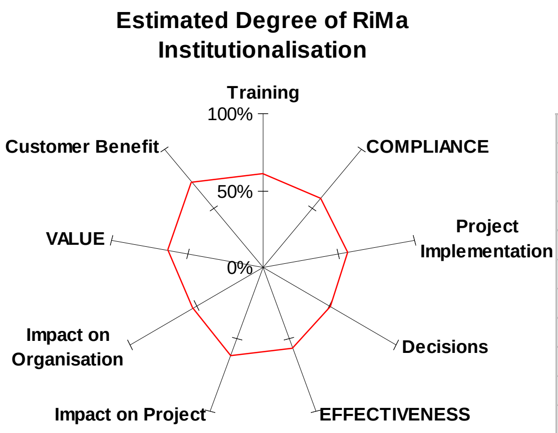
The rest of the projects fall somewhere between the above examples.

Lessons Learned

- Project leader's feeling and understanding of the process are keys to success.
- Starting too wide (in terms of number of triggered risks and actions) will almost guarantee failure.
- Each risk and the corresponding actions need a proper level of escalation and reporting.
- Calculating ROI (Return on Investment), in spite of some courageous trials, was next to impossible, given the required tracking that was not always in place.

Corrections to the Process

Realising that RiMa process was facing difficulties, SPI Management agreed to organise a tour through the main European locations in order to collect the experiences in the initial process institutionalisation, to try to understand the difficulties and to collect the suggestions for improvements. We assessed the degree of institutionalisation that RiMa had reached. After doing so, the management and the RiMa Council came up with the improved process. The improvements are summarised in the following subsections.



Training	61%
COMPLIANCE	58%
Project Implementation	56%
Decisions	50%
EFFECTIVENESS	56%
Impact on Project	61%
Impact on Organisation	53%
VALUE	62%
Customer Benefit	72%

Figure ZIVKOVID.3: Estimated Institutionalisation Level

Improved Understanding of Scoring System and Expected ROI

The scoring system was based on 3 simple values for impact (high, medium, low) and probability given as percentage [0-100%]. We have learned that each participant in the process had his own understanding of the meaning of the 3 (high, medium, low), not necessarily similar to that of the others. More embarrassing was communicating the scored values to managers; that led often to unnecessary discussions where nobody could prove or explain anything. Also the percentage as the means to express probability seemed to be to detailed.
For those reasons, probability was also described in terms of the simple (high, medium, low) categories, and the 2 extremes (negligible, almost certain) were added for completeness.
Impact on each of the possible categories was described in a document called "RiMa Impact Model", giving a firm, tabular definition to the (high, medium, low). The possibly impacted project category "climate" was replaced by "commercial cost". Similarly, categories "difficulty" and "cost" were defined to allow scoring the proposed mitigation actions. Those definitions are supposed not only to

- largely line-up the understanding of PRiMaTs, and
- make their communication towards the management straightforward and understandable,

but also they

- serve as an up front estimation of the savings that **might be** achieved if the risk would **really** pose a threat to the project.

The final evaluation of the RiMa performance and likely-to-be achieved ROI are left to the project final ("post mortem") review. No further questions asked.
This approach was piloted in 3 projects throughout the next half a year and accepted for wide use lately.

Finding the Right Level of Escalation and Reporting

Our experience, in several projects, showed that there were cases where project teams were dealing with rather less critical risks, mostly technical ones. Real managerial risks were left intact until they materialised as problems. More dangerously, those were many times not reported to the management in charge.
This behaviour was possible due to the following reasons.

- Agreed criterion for prioritisation of "Top 10" risks. The prioritisation of the identified risks took place based on 2 different criteria, i.e. "risk exposure" and "risk reduction leverage". The first was used as an intermediate product, based on probability and impact, to judge which risks were possibly the most damaging ones. The second took also the identified actions into consideration and was a measure of the achieved leverage if the actions were successfully executed. Both theoretically and practically, some of the important risks with high exposure would never appear on the "Top 10" prioritised list due to inadequate or impossible to define actions.
- Reluctance from some project teams to report on risks. Due to prevailing state of the mind that RiMa was "an internal project business" and due to some misunderstandings during some of the project reviews, some of the projects simply stopped reporting on RiMa.

We have worked both with managers and practitioners to overcome this.
Managers were explained more details about scoring principles. This was also

combined with explanation and build up of the previously mentioned impact model.
With project staff we worked through our coaches. We have agreed to monitor carefully the difference between the intermediate “Top 10” risks list based on “risk exposure” and the final one based on “risk reduction leverage”. For the items that were dropped from the intermediate list (and consequently finished on 11th or more position) we have agreed to make a separate list that should be shown during the project reviews with management. The most important item (possibly two or three of them) would get more detailed explanation. In this way the toughest items would be calling for the management attention and a decision on how to treat them, or possibly ignore them, would be taken at the appropriate level and shared among the stake holders and project teams.

Synchronising of RiMa Reporting

The reporting was largely experienced as a matter of taste. The risks presented took different forms, included different parameters, with or without actions to mitigate the risks. This was sending confusing messages to the management and might have discouraged some, especially middle management, to consider seriously reporting coming from RiMa activity.
Answer to this problem was to standardise the form in which the “Top 10” were reported. The table ideally fits on 1 page, includes a selected subset of parameters related to the risks that are of interest for the management or wide audience.
In spite of this agreement, this had remained largely a matter of personal understanding and taste.

Current Status of the RiMa Process Implementation

With these adaptations RiMa process continued through the next 9 months. Currently over 40 projects across 12 locations are involved.
Though it is difficult to precisely make a distinction and calculate to which extent each of the SPI initiatives contributed to the overall improvements achieved in the field of software engineering in the organisation it is believed that RiMa has played its role. We have achieved the short term goal, i.e. improved project management. We have shifted the focus of project managers and their staff towards future. We have taught our people to approach the project more often from the customer point of view. As a side effect, we have identified a number of process improvements in the areas other than project management. They will further optimise our processes and focus our attention to the benefit of the end result.
The shared opinion is that as general statement stands that SPI is about the change of the organisational culture, even more so is true for the risk management. So the state of the application of this process goes hand in hand with the maturity level we are experiencing, even when we are trying to take some lead. In this respect, several initiatives are being taken.

Next Steps

More Involvement of Local Management and Experts

We have observed that identification of risks remains somewhat limited to the professional scope of the people involved in the identification subprocess. The quality of the actions defined within mitigation and contingency plans may suffer similar lack of creativity. And, obtaining more co-operation and common understanding from the management may also require some more engagement on both sides.
To make further progress in this direction we have started a pilot of the slightly modified process.

- Identification: In addition to the project staff and other project participants, we have included also other sources of possible risks, some of them being outside our RD&E organisation (Figure 4). For example, these include interviews with commercial staff, management of Product Strategy group, etc.
- Risk Analysis and Prioritisation: The expectation is that a dedicated team of practitioners, experts in their fields, might be better positioned to propose more suitable actions for mitigation and contingency plans.
- Action Planning: For the items with larger organisational impact, like cross-project or cross-location risks, it is important to obtain support from the responsible levels of management. Therefore, the regular Steering Board of the location involved will discuss those items, (re-)confirming the commitments to the plans proposed by expert teams.

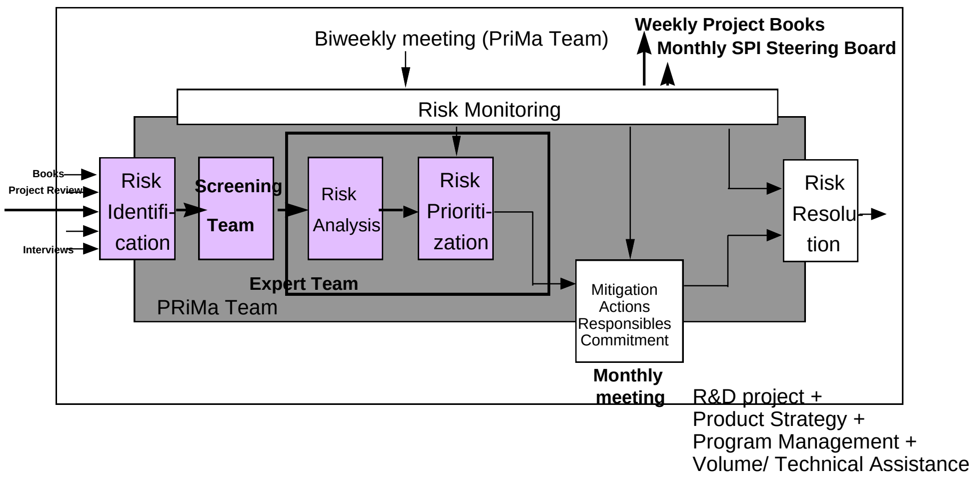


Figure ZIVKOVID.4: Improved Risk Management Process

RiMa for Requirements Management Through Technical Tendering Co-ordination

The SSD RiMa process as defined so far, urged for this activity, as a specific and explicit project activity, from the start of TLD onwards. Requirements analysis, allocation, project planning and resource allocation are handled by different groups of people. How the information related to the project preparation that was collected by those people was passed to the project teams, was not sufficiently defined and practised.
A joint effort is being made by the members of Technical Tendering Co-ordination

teams and RiMa Council in order to channel this information, in many ways relevant to the risks existing for the project, to the PRiMaTs, in a consistent and coherent manner.

RiMa Across Several Projects

In a complex environment such as that of SSD S12, major risk of any single project is impact due to some other project(s) running in parallel, or decisions taken for some previous projects of the same customer or market. In order to better manage this kind of risks, we have introduced what we call a Cross Project Control Board (in analogy with software configuration management boards).
The members of this board are the key people in PRiMaT/SQA of the current project and TPMs/SQA of the other projects that may possibly be impacted. The task of these people is to identify risks and problems for those other projects born by the actions and decisions taken in the course of the current project, and propose mitigation or contingency plans. Those are further discussed by the Steering Boards and the main decisions taken: either to deal with the items immediately during the ongoing project or to postpone the actions and pass them to the other project(s). The second may refer to the first coming project, or distribute among several of them.
The identified problems remain registered for follow-up in the defect prevention database of the impacted projects. This can be either location specific or a general one. The identified risks are passed to the PRiMaTs of the impacted projects.
Sometimes is the difference in this classification (problems versus risks) an academic one. This is specially the case for the cross project items. This duality can best be explained by the figure 5.

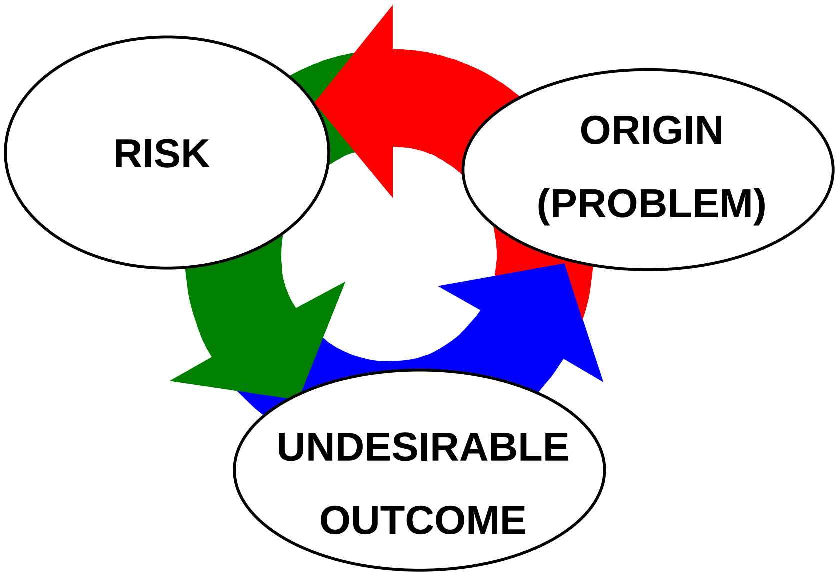


Figure ZIVKOVID.5: Spiral of risks and problems

Conclusion

We have illustrated an example of the institutionalisation of the risk management process into a large and complex software organisation. The process of risk management has proved its role as the improved way to conduct project management.

As already known any SPI initiative consumes considerable amount of time and effort. It changes organisational culture. This change is necessary in order to allow the organisation to make technological break through and learn how to develop new products.
Risk Management is even more a matter of cultural change. It is supposed to make forward looking habitual behaviour. As such, the process goes hand in hand with managerial and engineering practices that are described in CMM for software, for the organisations operating at levels 2 and 3 of the Model.
For this reason it may be expected that full benefits of this process may be expected after those practices have been well established. Indeed, our observation is that for the most advanced projects where enough priority and attention were given to the full range of SPI practices, risk management was also better established and the projects were more taking advantage of it. Risk Management Council in SSD remains taking the leading role in adjusting the process continuously (tuning where necessary and making advances where possible) while it is observing and estimating the process feasibility and utilisation.

References

[1] R. Charette: *Software Engineering Risk Analysis and Management*, McGraw-Hill, New York, 1989

[2] SEI (6 authors): *Continuous Risk Management Guidebook*, Carnegie Mellon University, Pittsburgh, 1997

[3] R. Charette: *Risk Entrepreneurialism: A Study For Alcatel Alsthom*. ITABHI Corp, Springfield VA, 1996/05.

[4] Bezirgan, Atilla; Graef, Nikolaus; & Mulazzani, Marco. *Risk Management. Alcatel*, 1993/04/30.

[5] GRafP Technologies inc. Software: *Process Risks Identification, Mapping and Evaluation Resolver*. 1996/04.

[6] Smith, Graeme. *Common Information Model Project Risk Management Guidelines*, Alcatel Alsthom, 1995/04/28.

Improving the System Test Process

Jörn Münzel
Bosch Telecom, Frankfurt, Germany

Abstract

The paper shows the results of an ESSI-funded experiment which has evaluated the use of a standardised test programming notation to increase test efficiency at Bosch Telecom. The goal of the experiment was to decrease system test time and to decrease manual effort involved in maintaining regression test cases through automatic testing.
The experiment includes changes of the test process based on the integration of a test case design and programming method together with the installation of new tools. Beside the experiment the paper describes the results and the experiences made with the approach regarding costs, effort and organisational aspects.
The main experiences of the project show that the increase of automatic testing is possible, but it requires a close binding to product development and new skills of testers. Efficiency in the use of test programming, which was not reached in the experiment, needs a high degree of test case reuse. Further activities to improve the approach based on an easily programmable and easily maintainable test design are also outlined.

Introduction

System testing, i.e. the functional testing of complex embedded software in a target environment, is often a very time-consuming and expensive task. **Bosch Telecom** develops and manufactures, among others, large private communication networks, which are currently tested by a mix of manual and automatic test sequences.
Looking for ways to improve the existing test process, Bosch Telecom decided to evaluate the use of the standardised notation **TTCN** (Tree and Tabular Combined Notation - ISO/IEC IS 9464-3), together with currently available test environment tools. This evaluation is embedded in the **ESSI**-funded Process Improvement Experiment 'RESTATE'(REuse of System Test cases through Applying a TTCN Environment - PIE-No. 23978).
The main goals of the experiment are test effort reduction and a shortening of the time spent on performing system test.
The paper presents an outline of the experiment (old vs. new test process) including the expected goals in detail and the motivation for introducing TTCN.
The measured results and the assessments made are the base for further activities to

disseminate the technology in Bosch Telecom together with further improvements in test technology.
The paper also presents experience about the Bosch Telecom approach to technology transfer and process improvement. Bosch Telecom has established a centralised **'software best practice' competence team** (called Software Technology Department) which has the task of improving software development by collaboration. Members of this team together with testers from the development department have carried out the experiment project.

Outline

The paper is structured into four chapters followed by a short conclusion.
The first chapter **'Process Improvement Experiment'** describes the context of the evaluation experiment including the goals, technical aspects and organisational remarks.
The second chapter **'Improvement Activities'** then shows the changes applied to process, test technology and the organisation.
The third chapter **'Experience and Assessment'** expresses the results of the experiment with reference to costs, effort and development culture.
The fourth chapter **'Future Activities'** summarises the current dissemination activities and the additional activities to improve the TTCN test technology.

Process Improvement Experiment

The chapter will give the reader information about the context and the goals of the improvement experiment together with some technical details of the process models, technologies and tools which were used.

The RESTATE experiment has had the goal to evaluate the use of a standardised formal notation to specify automatically executable system test cases. The supporting tool environment is used to perform test case execution in different environments (target and development environment) in order to reuse test cases. Test objects are private communication switches on special hardware under real-time conditions.

Bosch Telecom

Bosch Telecom forms the telecommunication business sector of Robert Bosch GmbH. It is concentrated on communications technology for public and private networks, and mobile telephones, as well as on security and traffic control systems.
For further information see Appendix II or connect to URL: ‘http://www.bosch.de’.
The products concerned are mostly developed in-house, with an increasingly heavy emphasis on software, sometimes in excess of 80 %.
Market openness, technological variety and rising customer expectations are forcing vendors in the field of telecommunications to come to market faster with high quality, better tested products.

Software Technology Department

To improve software processes and software development Bosch Telecom established a software technology department some years ago. The goal of the department is to evaluate ‘software best practice’ technologies and to integrate the proven ones into Bosch Telecom practice. Such technologies may already be in use in some parts of Bosch or they may be completely new.
To do evaluation and integration the department works very closely together with the product development departments. New technologies normally are evaluated in a prototype project in a realistic context and with the participation of the department which currently wants to use the new technology first.
To transfer technology, the members of the department work as consultants, teachers and coaches. The main work and basic strategy is to work as participant coaches. This assistance normally continues for a longer term and may last for the complete duration of a project. Main advantages is that the people being coached are in close contact for a longer period with the coach and the ideas of the new technology. Additionally the ‘expert’ and the new technology have to succeed in ‘reality’ which increases the qualification of the technology and the experience of the coach.

Baseline

The division where the PIE is being carried out develops private networking equipment, mainly private switches and terminals. It has had a BOOTSTRAP assessment which indicated a very good level of maturity but however indicated also some points for improvement. One of these was test methodology and test automation.
The software development process is well structured in different phases, derived from

the German V-model, with several fixed product quality evaluation points.
Test phases cover several levels of quality assurance, e.g. unit test, software integration test, system integration test including such activities as feature testing (also known as system testing), load testing and field testing.

Starting scenario:
The goal of system testing is to check the functionality of the product in the target environment under real time and real usage conditions. The functionality of the private telecommunication switch consists of more than a hundred features concerning the connection and administration of asynchronously acting 'users'.
The system test consists of testing each feature as a single capability together with testing the correct correlation of related features. Tests are specified as textual descriptions of behavioral scenarios involving several users, e.g. 'participant A takes handset off hook and dials 4711'. Test execution is done by using real terminals to stimulate the test switch as described and by checking the system reaction.
The first step in test automation is done by capturing the test execution at the signaling interface and by replaying these test runs automatically. During the replay the signals (messages) of the system under test are compared with the recorded one. This possibility is used to do regression tests at the system test level of new product releases.
The main problem with captured test sequences is their lack of robustness in the face of small signaling changes and optional concurrent behavior of asynchronous working links. This sometimes forces new capturing phases after small changes.

The **current test process** is structured into six phases/activities which are listed below to show what effort was measured:

- Test Specification Design:

Refinement of the informal behavioural test scenarios with the information needed to execute and assess the test cases.

- Recording Test Cases:

Manual execution of the test cases and capturing the data at the signalling interfaces.

- Verification/Adaptation:

Analysis of the captured data and manual adaptation of the scripts where necessary (i.e. date and time are 'don't care'-values).

- Test Environment Preparation:

Installation and configuration of test equipment and test object.

- Test Execution:

Automatic re-run of captured test cases.

- Result Analysis:
  Comparison and assessment of the executed test case traces with the reference data (captured traces), especially when the execution shows differences.

Goals

The goals of the improvement experiment were derived from the hypothesis that the time it takes to do system test seems too high. This means the time between developing a new feature or new product variant and delivering the tested product has to be shortened to get better reaction times in the dynamic telecommunication market.
The two main goals focused on were:

- reduce time to perform system test
- reduce manual test effort

Derived from these goals the target of the experiment was to measure and assess the effects of the use of TTCN to increase test automation and to reduce manual maintenance effort.

The following paragraphs describe in more detail the relation between the goals and the realised experiment.

When analysing the existing test process, it was detected that current test automation via capture/replay has two main disadvantages. Firstly, a captured signalling sequence does not cover all possible correct dynamic behaviour of the system under test. To test the functionality of the switch does not necessary force a special sequential behaviour at the different signalling interfaces each time. Secondly, each captured test contains a complete sequence including all the used signals/messages which forces the storage of a lot of redundant data. Changes of message data have to be edited or newly recorded for each test case because no building mechanism using data references is possible.

To improve the process it was decided that a ‘test programming’ technique is needed to increase the flexibility of test sequences and to decrease redundant data.

Existing test programming techniques in the area of communication testing are home-made or related to a tool supplier or based on TTCN. When realising this, the different arguments of costs, own development and maintenance cost, efficiency and future portability were considered carefully. It was decided to use TTCN and buy existing tools because of the long term cost aspects and the higher portability outlook. Home-made or special supplier solutions seem to be more efficient, but not on a long term view.

Experiment Facts

The following part will give a short introduction into the basics of TTCN and the working packages of the experiment.

The **Tree and Tabular Combined Notation** (TTCN) is standardised by ISO as part 3 of the ISO/IEC 9646 IS (Conformance Testing Methodology and Framework) [1] and includes a **formal notation for specifying test cases** as sequence trees of message interactions [2],[3],[4].

The main feature of the standard is the use of **PCOs (points of control and observation)** to define *abstract* test suites (ATS). To stimulate and check a system under test, the tester has to define one or more test points (PCOs) and has to specify the stimulation and checking of messages as abstract commands.

To specify asynchronous or optional behaviour, TTCN offers features to define alternative receives, default behaviour and concurrency. To check time dependencies, there are commands to start and stop timers together with commands reacting on time-outs. To assess the resulting behaviour, each path of a message sequence has to be assigned a **test verdict.**

The experiment was structured into three work packages which were sequentially [5].

Work package one was entitled ‘Installation, Education and Preparation’. It comprised the evaluation of the available TTCN tools including installation of the chosen one. The participants were trained to the use of TTCN and the tools. Main task of the work package was the conception of the used new test process based on a new test design and programming method, named the Bosch Telecom TTCN modelling technique.

Work package two was called ‘System test at target environment’. It comprised of the realisation of two TTCN test suites including their verification and execution in the target test environment. An overview is shown below in the figure, more information about the test process and the tools is described in the following chapter.

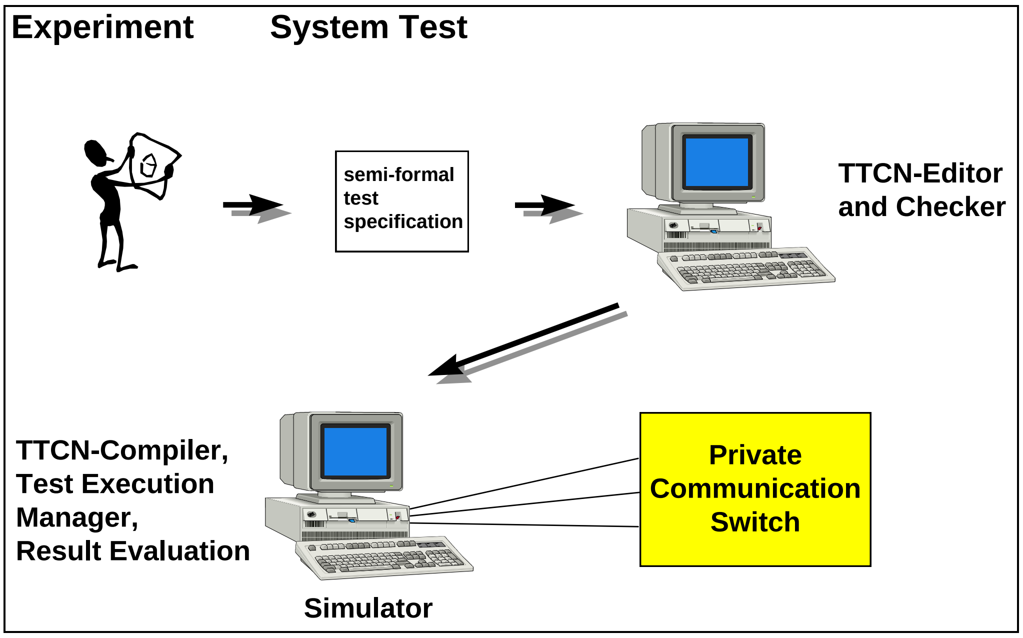

Experiment
System Test
semi-formal test specification
TTCN-Editor and Checker
TTCN-Compiler, Test Execution Manager, Result Evaluation
Simulator
Private Communication Switch

Fig. JMUNZEL.1 : Experiment Overview - System Test

Work package three was called 'Integration test with reuse' which evaluates a continuing improvement possibility. It contains the reuse of the realised TTCN test suites to test the switch software already in the development environment.
This part of the experiment is not included in this paper because these activities were not finished at the time of writing.

All the activities were measured based on a GQM (Goal/Question/Metrics) measurement plan. This includes also a baseline measurement where the same tests were realised and executed based on the current test process (see Baseline).

RESTATE Team

The project team consists of six members, three from the product department and three from the technology department. Additional a TTCN consultant from a German software house and a experienced process consultant were engaged to the project.
The main task of the technology department was the evaluation of the TTCN tools and the conceptual designs. The members of the product department mainly worked on the test contents, the baseline measurements and the test environment adaptations.
Regular project meetings, conceptual discussions, reviews, feed-back sessions and overlapping tasks encouraged a high degree of teamwork.

Improvement Activities

The following chapter presents the things which were changed by the experiment. This includes the test process based on a more formalised test method and the effects of using test programming techniques. Additionally the used tool environment and the organisational changes are described.
The idea of this chapter is to give some information about the activities done and technical results achieved in the experiment.

Process

Changes to the existing test process were forced by two aspects. One was the use of test case programming instead of recording, the other was the introduction of a new test realisation method to increase reuse of test data and behaviour.

The new TTCN test process is structured into seven phases which are listed below:

- Test Specification Design: (same as Baseline)
- Test Case Design:

Design of each test case structure based on the domain test architecture which forces the reuse of standard sequences and data.

- TTCN Coding

Coding of the test cases by implementing only the necessary new TTCN code and reusing existing parts (TTCN test steps, message constraints, etc.).

- Verification:

Analysis of the implemented TTCN test cases via code inspection and/or special test execution.

- Test Environment Preparation:

Installation and configuration of test equipment and system under test, compilation and configuration of the TTCN test cases.

- Test Execution:

Automatic execution of the compiled test cases.

- Result Analysis:
  Tool supported analysis of the executed test case traces, especially when the execution has a ‘failed’ verdict.

Technology and Tools

The use of **programming techniques** for test cases logically implies the existence of program development phases such as problem analysis, architectural design, detailed design, coding rules, etc.
In our case these activities resulted in a domain specific architecture [6], where each ‘user’ involved within the feature execution is modelled as a PCO. A user realises the interaction with a kind of terminal including telephone handset, display and keyboard. Each type of user is specified as a finite state machine (FSM) where each transition is designed as a logical building block. Synchronisation between the users and final verdict assignment of the test result are modelled in a central test task.
Because the used PCOs were not accessible with an automatic test environment each transition had to be transformed into a TTCN test step at signalling interface level, e.g. ISDN protocol messages. To reuse test steps they were coded as configurable macros. Problem of the representation is the not always clear relation between user

behaviour and signalling interaction.
Basically the structuring of independent, but synchronised test points including reusable test steps allowed an efficient specification and coding of test cases.

The **tool environment** as shown in figure JMUNZEL.1 covers three areas of TTCN test support:

- test coding;
- test compilation and configuration;
- test execution and result analysis.

Test coding is supported by a **TTCN editor** which allows the window-based writing of TTCN test suites, supported by syntax and semantic checks (online and off-line).

Compilation and test case configuration are supported by a **TTCN compiler** and a **PIXIT editor** (Protocol Implementation eXtra Information for Testing). The compiler needs to be specialised to the execution environment. The PIXIT editor is used to connect TTCN variables to the configuration data of the system under test, e.g. telephone number, hardware addresses. PIXIT data and executable code have to fit.
Test execution and result analysis are supported by a **test campaign manager**, a **PCO platform**, a **tracer** and a **TTCN animator**. The test campaign manager supports the selection and execution of single or connected test cases. The PCO platform is needed to support each used PCO with the underlying services. It realises the physical and logical access between the system under test and the test point. The tracer stores the execution data exchanged at all interfaces including time stamps and offers these data for further analysis. The TTCN animator is a tool which supports the trace analysis via showing the used path high-lighted in the TTCN code.

Organisation

The organisational changes affect two areas, one is the testers knowledge and training, the other is the project scheduling.
As mentioned before test case programming requires software development skills in addition to domain know how. That means TTCN testers need training in the use of TTCN, the tool environment and the Bosch Telecom TTCN modelling technique. Because of this amount of special knowledge, it was decided to build special test teams.
Project management is also involved in the changes because test case programming (at least initially) increases the effort for test specification and coding. These activities should be done in parallel to the product development to get executable test cases when the system test phase begins. Project scheduling and resource management have to be improved to integrate these changes.

Experience and Assessment

This chapter gives an overview of the measured results of the RESTATE experiment, the personal experience of the project members and the assessments made.
The information is structured into three sub-chapters to separate the aspects of costs, improvement / changes and additional factors.
This chapter should be a must for all readers because it contains the results of the RESTATE experiment.

Costs

The costs of introducing TTCN as a test programming technique were measured for three areas:

- tool investment;
- training;
- initialisation.

The **tool investment** comprises the TTCN editor and the TTCN Module with its parts described in the previous chapter. There was no investment necessary for basic hardware/software (PC, workstation, LAN, test simulator).
One licence for the TTCN editor is approx. 5,000.- US $, a licence of the TTCN Module is about 27,000.- US $. Assuming that each tester involved needs a TTCN editor licence and 3 - 5 tester share a TTCN Module environment, each tester's place of work costs approx. 12,000.- US $. In our experience, this investment is similar to a software developer costs (CASE tool, programming environment). Cost may decrease by negotiation and the number of licences bought. Additional effort/cost is necessary for tool installation and permanent support.
The **training costs** are distinguished between TTCN training, tool training and test domain training.
A standard TTCN training course (3 days) in our case was about 1,200.- US $ for each participant. The training included a 'hot-line' support over three month.
The tool training consists of a three day course and was about 2,400.- US $ each participant (3 participants). Tool support during the experiment without additional cost was part of the licence fee.
The test domain training depends on the knowledge of the tester about the features under test, the system under test and the interface specifications of the test points (e.g. ISDN protocol). In our case the cost ranged between no cost and 6 weeks effort for reading and coaching.
We assume training costs are also similar to software development, maybe less if the tester already has knowledge of software development in general.
The **initialisation cost** contains the effort we invested to develop our test architecture and our test design technique including TTCN coding rules and basic test steps. In the RESTATE project this effort was about 8 man months.
We assume that these costs have high dependency on the complexity of the test area (number of parallel test points, complexity of the interface protocols), the quality of an existing test architecture and the quality of the test case specification documents. In our case we had two complex interface protocols, up to four test points and no usable test architecture.

Effort

To get data for assessing the value of test programming techniques the experiment has measured the implementation of two test packages. Both test suites were implemented using the current test process and the TTCN test process. Additionally the test suites were adapted to a second type of switch to measure the change effort (maintenance) involved when reusing existing test cases.
In detail we measured the effort needed at each phase of both processes to get information about the entire effort and the effort distribution across the different activities. One test suite, called Basic Call, tests the feature of connecting two users under several conditions. The other test suite, called Advice of Charge (AOC), tests the feature of displaying and storing charging information.
Because we expected an effort increase for the first time programming a test suite rather than simply recording test cases, we also tried to measure the maintenance effort of both the processes. This was measured via testing two different switch types.
The following figures show the measured data of the 'Baseline' and 'Experiment' projects.

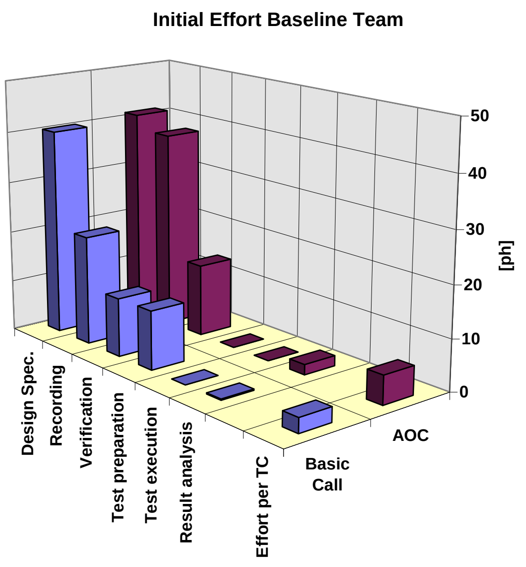


Fig. JMUNZEL.2 : Baseline Measurement - Test effort distribution

The figure above shows the results of the reference measurement (baseline) based on the current test process. Average effort per test case is 2.5 ph (person hour) for Basic Call and 5.6 ph for AOC.

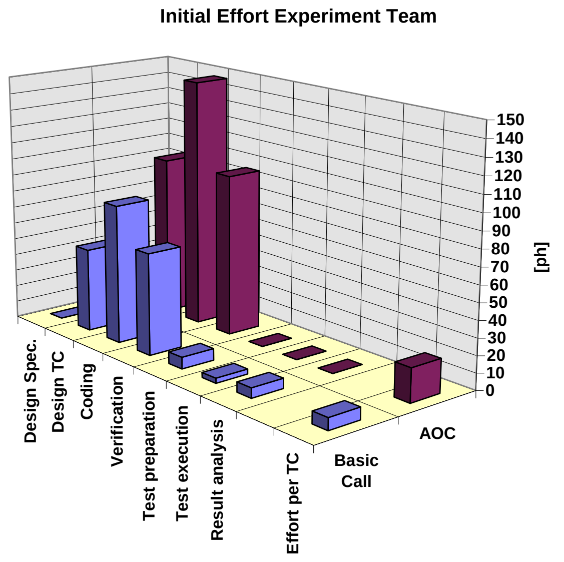


Fig. JMUNZEL.3 : Experiment Measurement - Test effort distribution

Looking at the experiment team, we measured the distribution shown above for the initial coding and verification of the two test suites. Average effort per test case (TC) are 7.2 ph for Basic Call and 20.1 ph for AOC. The zero effort during the 'Design Test Specification' (Design Spec.) phase is because we already used the results of the baseline team.
To illustrate the effects of using programming techniques and the reuse of building blocks in TTCN programming the following two figures show the evolution of effort per test case over the experiment period.

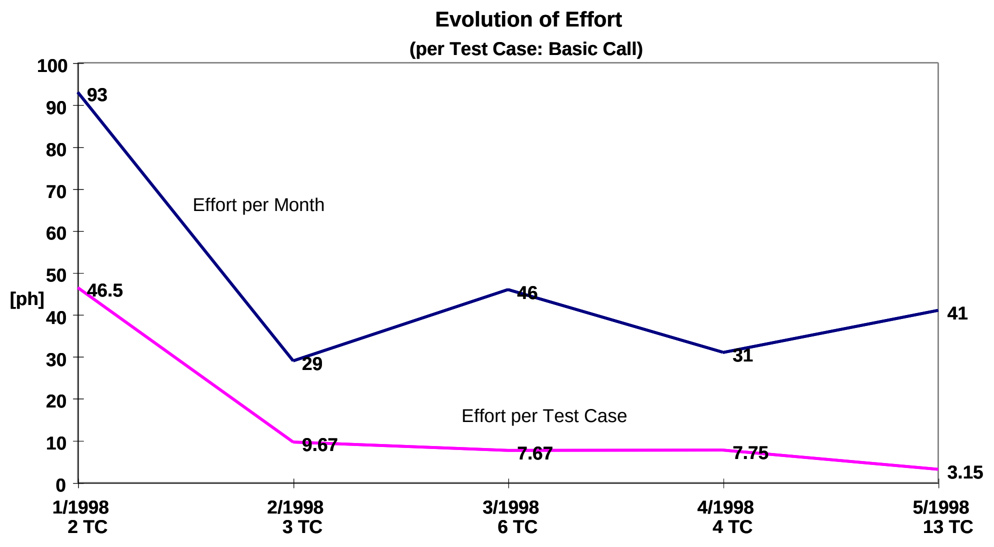


Fig. JMUNZEL.4 : Experiment Measurement - Effort per test case Basic Call

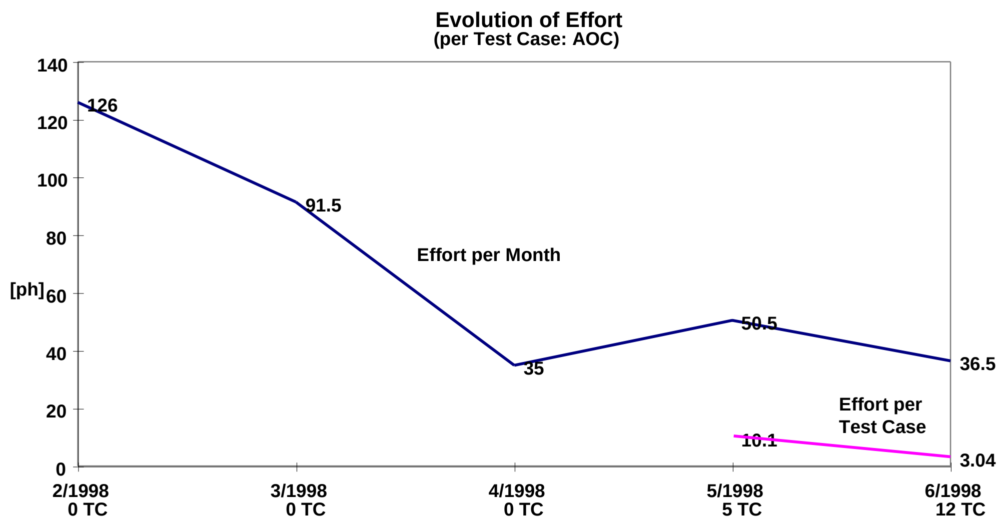


Fig. JMUNZEL.5 : Experiment Measurement - Effort per test case AOC

As shown above, the effort to program a test case decreases after an initial period in a range comparable to the baseline values.

The following two pictures show the results of measuring maintenance effort. We called this measurement the ‘Robustness’ factor because of our demand is that regression test cases should not need maintenance without required changes.

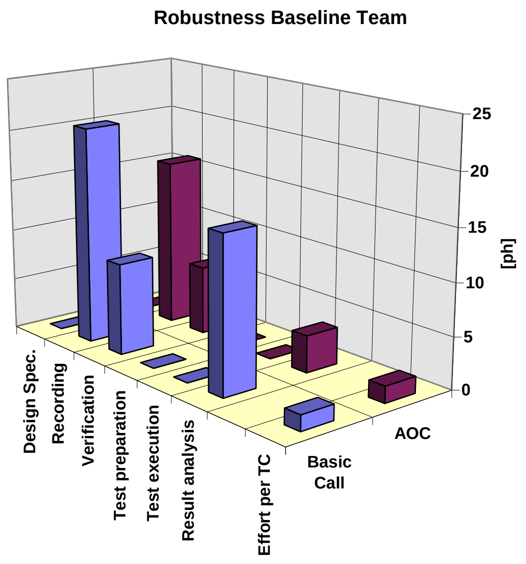


Fig. JMUNZEL.6 : Baseline Measurement - Maintenance effort distribution

The results of the baseline measurement show the activity distribution and an average effort of 1.5 ph for Basic Call and of 1.6 ph for AOC.

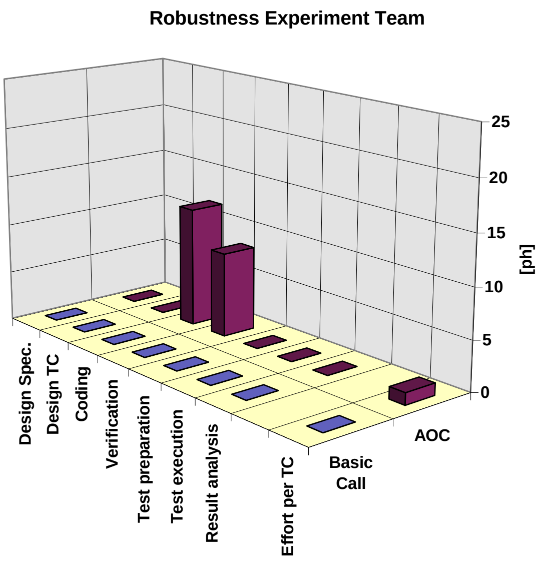


Fig. JMUNZEL.7 : Experiment Measurement - Maintenance effort distribution

The experiment test suite for Basic Call was already robust and the AOC required an average effort of 1.2 ph for each test case.

Assessing the **current test process** measurements we found that the main effort occurred during phase Test Specification Design, and that it was not expected at such a high level. The current test captures the test data at signalling level where a lot of communication details are included. Most time is needed to abstract from the detailed message contents (signalling level) to get statements about the behaviour of tested features (user level).
Another interpretation made is about the direct relation of test effort and complexity of test cases. The Basic Call test cases do have a less number of test actions than the AOC test cases to be specified, recorded and verified.
Assessing the **TTCN test process** measurements we came to appreciate the high amount of effort involved in developing / programming test cases. Until a library of basic test steps exists the effort is much higher. Although the experiment already used the Test Specification Design phase information of the baseline team, the designing and coding of TTCN took a longer time.
A positive evaluation of TTCN is during the Test Environment Preparation and Result Analysis phases where less effort was needed because of a higher degree of automation.
Interpreting the maintenance effort for both processes there are no striking differences between current process and TTCN. This may occur from the approach we made to test two different switches. For Basic Call we registered zero effort with TTCN and some effort with the baseline team, which shows the distribution we expected. For AOC we registered a higher effort with TTCN because the programmed configuration of one switch has differences to the second switch approach. This resulted in a small redesign of the TTCN test suite. Another influence factor we did not measure was the degree of comparison we used. In the case of RESTATE we checked the receiving messages very superficially and this was easy to adapt at the Baseline approach.

Detailed test checks should be easier to maintain in the TTCN programming environment than at the protocol simulator.
Concluding the effort data of the experiment we assessed that the expected advantages of using TTCN were not reached by the experiment. The goal to reduce test time and manual test effort are not deducible in a short term view compared to the costs and the effort.
On a long term view there are aspects which contain advantages and which we will use as a base for our ongoing work (see chapter Future Activities). One aspect is the effort reduction of TTCN programming for complex test cases based on a stable test step library and a lot of regression. Another advantage may be a higher degree of automation at regression test which may be executed at 24 hours at seven days a week. Another aspect where we guess an advantage is the possibility to do test case programming in parallel to product development which forces an early inspection of the interface specification. We guess that this would not only lead to earlier existing test cases but also to an increased quality of the realised product.

Management Experience

At current test approach there exists a high individual relation between testers, system under test and developer. The realisation of test runs and the interpretation of results is based on a lot of experienced know how. The easiness of the approach and the experience of the existing test team are useful to react fast on changes.
When using a test programming technique the designing and interpretation rules have to be formalised to be effective. The coding of stimulating messages and the verification of the receiving messages has to be based on formal specifications. In our experiment we had the problem of a mismatch between the level of testing (user behaviour) and the level of test access (signalling interface). Because a formal representation of user behaviour and the associated signalling behaviour did not exist, we had to build one. Such representation is required for programming test cases but was not directly accepted by the developer. From their point of view this kind of specification may handicap the flexibility of a layered development where realisation is hidden from the service interface (access point). During the experiment we could not solve this problem.
Another experience we had during the experiment is the similarity between the management of test case programming and of software development. In the current test process the testers do a lot of work independent of each other. When designing and coding TTCN the programmer have to work as a team to use common information, e.g. message declarations, constraints, test steps. To manage this work such technologies as team management, quality assurance and configuration management (access rights, release management, etc.) are necessary.

Transfer Experience

In the **RESTATE project** we summarise the technology transfer experience in three areas, technology evaluation, process changes and cultural / organisational changes
To introduce a **technology** like TTCN requires not only the tailoring of tools and naming conventions but also the development of a vision about a new process. We have spent a large amount of effort in discussing and understanding the ideas of TTCN based testing, the application domain context and the goals of the system test before designing a concept how the TTCN technology and tools may fit to the problem area. Successfully we early integrated together external consultants, experienced testers and members of the technology department, so that we got the

information needed and, in time, a common vision of the solution. Generalising this experience, we realised the necessity of adapting a ‘new’ technology to the problem domain. In particular the integration of experienced people of the application domain and the technology domain over a longer period supports a well designed and accepted solution.

The main changes in the test **process** were brought about by the increase of documentation depending on a higher degree of necessary formalism. Both sides, testers and developers, had reservations about writing and using documented specifications. On the other hand we got more acceptance during the project when we used our specification for problem localisation. The main problem of detailed documentation is often that filters are missing which could help in extracting currently needed information. Missing flexibility and the effort of maintaining detailed documentation often were used as arguments.

Generalising this experience we will have to think about techniques to get hierarchical structured specification to filter adequate levels of information and to maintain traceable and consistent data.

Our experience in **cultural and organisational** changes is limited to a small amount, because we did not work with a larger team over a longer time. In conclusion, we had no problem in discussing the problems and in designing a new test approach but there is still uncertainty as to how the results of the experiment should be interpreted. The problems of integrating a test programming technique are focused to two areas. The first is the need for a new kind of specification (representation ‘user behaviour’ - ‘signalling behaviour’) which has to be written by development staff. The second is the need to teach the testers how to program test cases. Finally also project scheduling should be changed to integrate a specialised TTCN test team early on in the development.

Because of the gravity of changes involved our doubts seem to be justified and we have to think about a stepwise integration.

Future Activities

Concluding the results of the RESTATE experiment the use of TTCN as a system test programming technology will not meet our process improvement goals in a short term view.
TTCN requires investment (tools, training, initialisation) and increases effort for the first test case realisation. Maintenance effort is less than today but to reach a return on investment we need a high number of maintenance cases.
On the other hand we learned a lot when evaluating TTCN and measuring our current test process. The main message for further activities is based on the experience that the developed test architecture including the use of PCOs is a strong concept to improve test specification and test verification. Also the TTCN technology with the existing tool environment is valuable to be used for test automation but the programming effort has to be decreased.
The following sub-chapter show an overview about some activities already started or aimed at.

Distribution

To distribute the results, experience and recommendations of the RESTATE experiment we started several activities.
Public dissemination of results and experience are done via ESSI reports and Conference presentations [5], [6] as this paper shows. In-house dissemination at Bosch is done via reports, presentations, intranet pages and several workshops.
Main distribution is supported by the ‘software best practice’ competence team which is involved in consultant and coaching activities. Some other development departments already started to include the TTCN technology to their process using the experience of the experiment and the know-how of the technology department.

Continuous Improvement

To improve the **TTCN test programming** technology we selected three areas for further activities. Main goal is to reduce the effort of test case coding.
We have started a co-operation with the Institute for Telematics at the University of Lübeck, Germany, to analyse the problem of automatic generation of TTCN test cases out of formalised Message Sequence Charts (MSC). MSC is a standardised notation to specify dynamic interaction on a higher level of abstraction. This should decrease the effort for TTCN coding.
Another area is to increase the use of formal specification techniques as SDL and ASN.1 for interface specification. These could be used for increased tool supported code generation at development and at testing.
A reduction of development cost per test case should be reached by increasing the reuse of test cases. This is the goal of the second part of the RESTATE experiment where the already existing TTCN test suites are executed during the software integration phase in the development environment (see Experiment Facts: Work package 3).

To improve the **test process** we started a further analysis about the current used test scenario (protocol PCO). Our target is to deduce PCOs which are easier to handle from the viewpoint of feature tests. Preconditions are a necessary access at this interface and a less complex interaction model. Additional we believe, that a stable syntactical and semantic interface is a basic requirement for the efficient use of a test programming technique. This will increase considerably the use of building blocks and reduce maintenance effort.

Conclusion

Summarising the results and the experience of the RESTATE experiment:

- TTCN as a test programming notation is only part of the solution for our test problem.
- TTCN, its test method and existing tools offer a mature basis for systematic and automatic regression testing.
- Efficient automatic testing requires a binding to the development process.

The best feature of the TTCN test method is the idea that tests are structured using Points of Control and Observation (PCOs) which supports a clear approach.

Efficiency of programmed test cases is reached through a high degree of regression to reduce maintenance effort. Efficiency of programming is reached through the use of stable interfaces as test access points (PCOs) to reduce coding effort. Interlocking the test process with the development process (e.g. requirements, architectural design) will support product quality. An earlier review of interface specifications from a tester’s point of view will increase transparency and completeness.

On the other hand, test case automation requires investment in tools, new skills, test process and initialisation effort.

We have not yet reached all our goals, but we are on a promising course for the future.

Glossary

AOC	Advice Of Charge (supplementary service of a switch to display charging information)
ASN.1	Abstract Syntax Notation 1 (Standard for specifying abstract data types)
ATS	Abstract Test Suite (TTCN test cases, independent of special execution information)
ESSI	European Systems and Software Initiative (Program of the European Commission)
GQM	Goal/Question/Metrics paradigm (basic metric concept, originated by Victor Basili, University of Maryland / USA, and the Software Engineering Laboratory)
IEC	International Electrotechnical Commission
ISDN	Integrated Services Digital Network (Standard for Telephony)
ISO	International Standardisation Organisation
ISO 9646	Open System Interconnection - Conformance testing methodology and framework (Standard including the specification of TTCN)
LAN	Local Access Network
MSC	Message Sequence Chart (Standard for dynamic behaviour flows)
PCO	Point of Control and Observation (basic concept of TTCN methodology)
PIE	Process Improvement Experiment (ESSI work program task type)
PIXIT	Protocol Implementation eXtra Information for Testing (Variables used in TTCN which are assigned to configuration data at run time.
RESTATE	REuse of System Test cases through Applying a TTCN Environment (Title of the ESSI PIE project of Bosch Telecom)
SDL	Specification Description Language (Standard for specification of finite state machines)
TTCN	Tree and Tabular Combined Notation (Test programming notation, specified at ISO 9646 - part 3)

Reference Summary

[1] ISO/IEC IS 9646, Information Technology - Open Systems Interconnection - Conformance testing methodology and framework, Part 1 - 7, *International Standard IS 9646*, ISO, Geneve, Switzerland, 1992

[2] Kron J., Wiles A., A Tutorial on TTCN, *Tutorial at the 11th International Symposium on Protocol Specification and Verification*, 1991

[3] Baumgarten B., Giessler A., OSI Conformance Testing Methodology and TTCN, Elsevier Sciences B.V., Netherlands, 1994

[4] Ek A., Grabowski J., Hogrefe D., Jerome R., Koch B., Schmitt M., Towards the industrial use of validation techniques and automatic test generation methods for SDL specifications, in: *Proceedings of the 8th SDL Forum ed. Cavalli and Sarma*, Elsevier Sciences B.V., Netherlands, 1997

[5] Münzel J., Better testing for private communication networks, in: *Proceedings of the 5th European Conference on Software, Testing, Analysis and Review*, Edinburgh, (CD-ROM), 1997

[6] Anlauf M., Programming service tests with TTCN, in: *Proceedings of the IFIP TC6 11th International Workshop on Testing of Communication Systems ed. Petrenko and Yevtushenko*, pp. 263-278,Kluwer Academic Publishers, Boston, USA, 1998

Appendix I : about the author

Jörn Münzel is currently section manager at the Bosch Telecom software technology department and project leader of the ESSI Process Improvement Experiment - RESTATE (REuse of System Test cases through Applying a TTCN Environment). The technology department propagates software engineering and methodology expertise through co-operation with the product development divisions.
Mr. Münzel is active in the areas of software test and software quality management.
From 1993 to 1995 he worked at Robert Bosch Research and Development as a software engineer in the areas of test and quality management of object-oriented software development.
Jörn Münzel received a diploma in Computer Science from the Technical University of Darmstadt (Germany) in 1986.
Prior to joining Bosch, he was a software engineer at a German software house, where he took part in and led several projects in the area of telecommunication and conformance testing.
For several years Mr. Münzel has been an active member of the German Special Interest Group on Test, Analysis and Verification of Software which is part of the German 'Gesellschaft für Informatik e.V' (German Computer Science Society).

Appendix II : about Bosch Telecom

Bosch Telecom forms the communication technology business sector of Robert Bosch GmbH. Approx. 19.000 employees develop and manufacture products in a number of locations in Europe which are marketed world-wide over the company's own sales network.
In 1997 Bosch Telecom achieved sales of approx. 5.3 billion DM which represents 11% of total Bosch Group sales. 560 million DM were invested in Research and Development.
The Bosch Communication Technology Business Sector is concentrated on communications technology for public and private networks, and mobile telephones, as well as on security and traffic control systems.
Six Product Groups are responsible for this business and are present in the market with a wide product mix. This ranges from systems for radio-relay, multiplex engineering, and network management, through telecommunications, fire and emergency-alarm systems, video-supervision installations, up to cellular phones, traffic-management systems and equipment for satellite engineering. The business is characterised by its strong service orientation.

Implementing SPI: Combining business process improvement and system development at Nokia

Tero Lindholm
Nokia, Salo Finland

A short description of Nokia

Headquartered in Finland, Nokia is a broad-scope telecommunications company supplying mobile phones, mobile and fixed telecommunications networks, data communications solutions, multimedia terminals and computer monitors. With sales in 130 countries, net sales totaled FIM 52.6 billion ($9.8 billion) in 1997. Nokia, listed on NYSE (NOK.A), employs more than 41,000 people worldwide.

The starting scenario

Nokia has rapidly moved from functionally oriented organisation to a business process oriented mode, and our information systems have not followed the change as fast as they should. To fill the gap, IM organisation must learn and implement working methods which enable the creation of business process oriented information systems. To satisfy these business needs, Nokia has introduced a new development approach for information systems development. These processes have been created and tested during the SPI.
In 1996, the starting point of software engineering practices in Nokia Mobile Phones/Information Management was, that there were no clearly defined

processes to guide the software development according to the business needs. Requirement analysis was usually done with an ad-hoc style without a pre-defined process. For the technical design and implementation, there was a first version of project manual to follow.
Technical environment from IT point of view at Nokia is great. Market's leading brands are utilized in every part of the company and personnel's skill level is high in technical issues. In the SPI project, ICL ltd was selected as the vendor to provide a consulting point of view and to deliver their skills for business analysis and requirements specification areas.
Business environment in telecommunication industry is fast moving and quite young, which means big challenges for information management: timing is crucial. Because of the changing environment, also organisation is changing rapidly. Change is an every day phenomenon at Nokia, which helps a lot when changing working practices as a part the SPI. Skill set needed for the business and requirements analysis and other parts of the SPI is different from the technical expertise so common at Nokia. In order to succeed, some time for training was reserved during the SPI.
The technical target environment for the SPI included a R/3 based standard package as the baseline project environment, a CASE tool to support the IM Process and an intranet environment where the results were shared.

The expected outcome

The expected outcomes of the PIE were as follows:

- Accepted Nokia IM process description, which
    - includes phases from concept creation to prestudy and planning
    - Is integrated to Nokia process development methodology
- Existing software support (a CASE-tool has been selected)
- Trained facilitators are available
- A pilot project ready
- Other projects has been started
- Vendors as partners to develop the process and the tool further.

These outcomes were also achieved during the project.

The implementation of the improvement actions

Organisation

For the target organisation, the main activity has been the Workpackage 1, which purpose was to sell the development ideas to the senior management of the Information Management organisation. Success in this step can be also seen as the most important success factor for the experiment. Senior management had the role of steering group and provided the management support for the project. The project was organised as follows:

The steering group consisted of senior management of the information management organisation, who are responsible of information systems in various groups of Nokia. The project leader came from the IM organisation as well. The project group consisted of IM resources, who actually do systems development as their daily work. These persons participated the project as a part time resources. Business organisations were involved through the pilot projects and the quality office was represented through the project leader, because he was a part of the team which was developing the process development methodology of Nokia.

Technical environment

The main technical outputs were the process descriptions and ARIS Toolset as the CASE-tool. The main learning point is that when the target process is clear, also system requirements are clear. And when system requirements are clear, it is quite straightforward to analyse and test the possible candidates.
The process of selecting the CASE-tool consisted of the following major steps: a) collect the selection criteria including information on vendor companies, functionality required and the technical environment the tool runs in. b) select the candidates roughly using the market knowledge in house and some external reports like the ones from Gartner Group. c) arrange workshops where the candidates are checked against our criteria d) select the Top 2 candidates and pilot the best one. e) If the pilot is a success, stop otherwise go to the step d). Vendor information is needed because Nokia wants to have partners we can trust in, the technical environment should fit into our current environment and the functionality requirements are derived directly from the process descriptions, e.g. if we have a step called “map the process” the tool must be able to support the mapping exercise.

Training

Training has been divided in three main parts based on the targets of the PIE: the focus areas were process thinking, IM methodology and the tool . Process thinking and development principles were introduced to the team at the beginning of the project, when the focus was on process descriptions. The IM methodology principles were introduced to the management when the first draft versions of the descriptions were available with purpose to show some early results. The pilot project teams were trained as a part of the project training in order to learn a common methodology and language during the development work. Aris toolset training was given to the PIE team when the tool selection process was over in order to enable the tailoring of the tool.

Role of the consultants

The role of the external consultants was to transfer their knowledge about IM development processes to Nokia’s organisation in terms of process descriptions. The transfer was made mainly through process descriptions and personal consultancy. At the beginning of the project, they checked the preliminary, rough process framework which had been developed in Nokia.

The purpose of the evaluation was to quarantee that the starting point and the targets were at a world class level. During the project, the consultants then fulfilled the framework with further details and descriptions. For example, when the original framework included an activity called “Analyse data subject areas”, the consultants then added the detailed descriptions about the activity to Nokia’s model. The detailed descriptions were mainly based on the existing process descriptions of ICL ltd.
Nokia expected to gain a lot of benefits by using consultants, because the selected vendor has more experience about the problem area than Nokia and they had also a tested their methodology in practise. That existing methodology is now transferred into Nokia’s language and it has also been fitted into the Nokia process model.
The tested methodology means here process descriptions, that have been created in ICL based on practical experience and that have been applied in many projects already. Translation into Nokia language is needed because the terminology is different in each company: for example the word “process” is defined differently and ICL uses the word “task” instead of Nokia’s “activity” etc. The tailoring part means that Nokia provides the framework of what kind of activities we need to perform during a development project and ICL is capable of providing the detailed descriptions of the activities: how to do an activity in practice, which roles are involved, what kind of information is needed and created during the activity etc. It has been obvious that the most of the activities Nokia requires can be found in different ICL descriptions, but they are grouped in a different way.
Experiences were mainly positive and the main target of why the consultants were used, was reached.

Phases of the experiment

The project was divided into phases according to Nokia’s new system development process. The main phases and milestones are described in the figure below.

ID	Task Name
1	*ESSI PROJECT (IS CONCEPT CREATION & PRE-STUDY)*
2	Milestone E-1 (PRE-STUDY START APPROVED)
3	PRESTUDY
6	Milestone E0 (PROJECT PROPOSAL APPROVED)
7	PLANNING
14	Milestone E1 (PROJECT APPROVED & DEFINITION FROZEN
15	PROCESS & TOOLS DESIGNING
62	Milestone E2 (PROCESS DOCS and tools APPROVED)
63	PILOT
85	Milestone E3 (PROCESS WORKS IN ACTUAL USE)
86	ROLL-OUT
95	E4 (PROCESS WORKS IN PRODUCTION USE)
96	FEEDBACK
105	E5 (PROGRAMME TERMINATION)
106	PROJECT MANAGEMENT

Fig. TLINDHOL. 1:

The model includes milestones where the main outputs of each phase are reviewed and accepted. Milestone E-1 is the starting point for the development; E0 handles the project proposal including the plan with main resources, steps and schedule; in E1, the process specifications and detailed project plan are accepted; E2 handles the created process descriptions and the selected tool; in E3, the results of the pilot project are accepted. At this point, also the Process Improvement Experiment (PIE) was over.

The descriptions were created using a piloting approach, where three draft versions were introduced. By using the drafts, we were able to provide quicker results to the organisation and also collect feedback at early phases of the project. Three draft versions seemed to be practically the optimum number of releases, because the timing and effort required to produce them was at an acceptable level compared the benefits gained. A top-down approach was used to develop the descriptions, which means that each draft release had more details than the previous one, and that each release included the descriptions of the whole framework.
The selection of the CASE-tool started when we had a rough understanding about the development process. This kind of approach made it possible to start the selection process at an early phase of the project in order to select the top candidates.

The measured results and lessons learned

Technical results

The following technical results have been achieved:

- Process documentation is available
- A tool (ARIS Toolset) has been selected and purchased to support the process described above
- The process has been piloted in pilot projects and the experience has been recorded in order to develop the process further.

The main measurement criteria for the technical results was time: the results must be available in the planned schedule. From the schedule point of view, the project was successful. The deliverables of the project have remained the same as planned at the beginning of the project. The project team has had many positive experiences concerning the methodology: some of the available documents are already in use in real life projects and many persons and organisations external to the pilot projects are interested in the results. One of the main targets of the project was to integrate process thinking and system development, and it seems to be obvious that the integration is really needed, and wanted by the organisation. In practice, the whole application development in Nokia is based on process thinking and now we have a methodology to support this approach.

According to the **SPICE standard (Baseline Practices)** , the current status of the development processes are as follows:

SPICE Process Category	Status at the beginning of the PIE	Status today
Customer-Supplier	Contract process exists, customer satisfaction is measured	No changes
Engineering	No process descriptions and proper practises exist for developing purposes, implementation is supported by a project manual.	All the needed process descriptions exist
Project	A first version of the project manual has been used since March 1995.	Updated to include the required guidelines and templates like project plan and review documents
Support	Quality assurance and problem solving has been used according to the project manual, also since March 1995.	Quality assurance included into the development process in the form of milestone practices
Organization	Engineering the business (ORG.1) will be tested according to this proposal, process improvement is	Engineering the business is a part of the development process (process improvement). IM related process training will be done as requested by the

	mostly based on outputs of that test. IM related training is well organized, and work facilities are at a proper level. However, reuse of application components is not supported.	projects, there are no changes in other trainings. No changes in work facilities. Reuse is now emphasised at the design level, when creating the system specifications.

The high level description of the development process includes the main activities described in the figure below.

ID	Task Name
1	**IS CREATION PROGRAM MODEL V. 1.0**
3	**IS CONCEPT CREATION PROJECT**
4	Plan Concept Creation project
5	C1 (START OF CONCEPT CREATION APPROVED)
6	**BUSINESS ALIGNMENT**
7	Analyze strategic needs
8	Define the process scope
9	Analyze current state of the process
10	Map the process at rough level
11	Analyze end-users' needs & ergonomics
12	Analyze interest groups' needs & effort
13	Document high-level business requirements
14	C2 (BUSINESS NEEDS SPECIFIED)
15	**IS ARCHITECTURE DEFINITION**
27	C3 (IS CONCEPT PLANNED)
28	**IT CONCEPT VERIFICATION**
41	**IS ENGINEERING PREPARATION**
52	C4 (IS/IT CONCEPT VERIFIED AND APPROVED)
54	**IS ENGINEERING PROJECT**
55	Plan target-setting phase
56	E-1 (TARGET-SETTING START APPROVED)
57	**TARGET-SETTING**
58	**ANALYZE THE PROCESS**
62	Determine people change mgnt requirements
63	**DETERMINE BUSINESS REQUIREMENTS FOR SYSTEM**
69	Make Non Disclosure Agreements (NDA) with vendors
70	**VERIFY AND DETERMINE SYSTEM COMPONENTS**
74	Compose project proposal
75	E0 (PROJECT PROPOSAL APPROVED)
76	**IS & PROJECT PLANNING**
77	Create system specification
78	Plan project
79	E1 (PROJECT PLAN & SPECIFICATIONS FROZEN)
80	**IS DESIGN & VERIFICATION**
108	E2 (SYSTEM WORKS IN TEST ENVIRONMENT)
109	**IS PILOT**
125	E3 (SYSTEM WORKS IN ACTUAL USE)
126	**IS ROLL-OUT**
138	E4 (SYSTEM WORKS IN PRODUCTION USE)
139	**PROJECT EVALUATION & COMPLETION**
140	Collect lessons learned
141	Compose final report
142	Close project data store
143	E5 (PROJECT TERMINATION)

Fig. TLINDHOL. 2 : List of activities in the Nokia IS Creation model.

The activities with name Cx or Ex are milestone activities, that act as quality control and decision points in each project. Each activity is described in detail using three different descriptions: *process description* describes the activity in a graphical process format telling of what should be done, *templates* are input and output documents of the activities and *guidelines* are detailed descriptions of how the activity should be performed. The selected software tool supports the process making it possible to describe and analyse the results of a development program in an electronic format.

Business results

The business impact of the experiment is mainly in the area of process development: the main pilot area, Service and Repair process, has been re-engineered according to the principles of the new development process. The main targets like reducing the process turnaround time and simplifying the system structure, will be reached when this actual baseline project is ready. But already now, when we have created a new process and the needed system specifications, the new development process has proved to be a valuable tool also from business point of view.
The real challenge is in organising a development program where both process improvement and creation of a new information system takes place. Management, program organisation and the users must all understand that the actual system implementation will happen quite late when counting from the beginning of the program. Early wins are important assets in these programs, and they can be achieved using package programs, demonstrations and piloting approach.

Organisational results

The main impact to the IM organisation is that the process thinking has been learned also in the IM community. The importance of business view in the system development has been realized and the application organisation has been organised according to the business processes. The PIE has had its impact to these changes, because the IM organisation has a development process to follow when developing applications. In order to quarantee that the internal IM process development will continue, a new job position has been created with responsibility of process development.

There has been a positive change in organisation culture in different areas:

- Project team understood the business purpose of this exercise, resulting in a good working spirit.
- Management of the information management organisation has been interested in the project results, which means that the development ideas has been bought by the management.
- Those who don’t know about the results of the project, seems to have intellectual resistance to accept the ideas. Anyway, they were asking the draft-results of the project at the earlier stages and now when they have the descriptions available they are also using the descriptions. This means that the demand is out there.
- From cultural point of view, informing about the results is very important. This challenge will be tackled by the new process improvement organisation and communication material.

The main achievement in skills area is among the project and pilot teams They have got a wider view to the work in information management and now they have a good reason to be interested in business development instead of only technical aspects. Although, the team was interested in business issues already before the project started, but now the target of skills development is clear. The team is also able to provide skills development programs for other IM persons, because there are general process and skill maps available. Some of the new skills which are required in IM are for example process thinking, architectural development, basics about human related change management and creation of a development roadmap.

Technical lessons

The following lessons have been recorded during the experience:

- Process principles drive selection of technology but also vice versa, technology may provide new business possibilities
- Business reasons for development must be clear in order to optimise the process and support the business
- Deep technical aspects are not as important as communication; modular and clear presentation about the working practices is needed at three levels as described in the table below:

Target group	Type of material
Management level, Newcomers	Overview presentation
Project leaders, team members while working	Practical easy-to-use working guides in a form of document templates
Project leaders, newcomers, team members in training sessions	Process description in a graphical format (describe what to do) and deeper technical guidelines (describe how to perform an activity).

Delivery and availability of the documentation is one of the most important topics when rolling-out new working practices. A clear, easy-to-use channel should be created for this purpose if not existing before.

Business lessons

According to the experience, the critical success factors for process development are the following:

- Active communication to keep the project alive
- Sponsor and management support to sell the ideas forward
- Business cornerstones (reasons) must be clear from the beginning
- Clear milestones enable the project follow-up and communication
- Quick wins must be achieved in order to show business possibilities of the project
- The process must be easy to understand, easy to learn and easy to support. This can be realised by handling it like a service product: a real product requires a product manager, customer support and continuos improvement based on real-life experiences.
- Resourcing is a critical issue: enough resources must be granted for developing the new process, training it and supporting new projects to get started.
- Training must include a lot of justification of why this kind of approach is needed. Selling of the ideas is a difficult task which must be done over and over again, in practice to all new process users.

Strenght ans weaknesses of the experiment

Strengths of the project were as follows:
- There users of the target process are very demanding, and they challenge the development work all the time
- Nokia is a process oriented organisation, which means that the developers can concentrate on developing the working practices, they don't have to sell process thinking to the organisation
- Open minded project group, who was able to listen to the environment and learn new things very rapidly. This feature was applied visible in the documentation, where new ideas were adapted quickly.

Some weaknesses experienced during the PIE:
- Delayed start of the experiment because of long processing time in European Commission (EC)
- Resource planning problems, partly because of the delayed start, partly because of other organisational pressures
- Too detailed reporting is required by the commission
- In the company, a part of the company culture must be changed by the project: methodologies are not highly appreciated in a fast moving business. This means that a lot of effort is required for communication.

If I were to repeat the experiment, I would change the following tasks:
- Start the project without EC funding; It would have been easier to gain organisation's commitment if the project had started as planned. Because of the delay, few key persons in the organisation had to start another project and they were not anymore available for the PIE. The support from the Commission is a great thing, but because time is one of the most important factors in telecom-business, delays in projects may result in difficult problems.
- Management support should have been heavier and more visible than it has been; I would require more visible actions by management to support the project.

Conclusions

Human related change management is one of the key issues in process development. Even though it was known from the beginning of the project, not enough manpower were focused on communication. The lessons have been learned and the future actions include a lot of training and other kind of communication.
Co-operation between Nokia and the partner has been a positive experiment. Because Nokia had clear specifications on business requirements and the process framework, the partner was able to start its work to specify and detail the framework process and use its competence in the required area.
It is clear that the results of the project have been very good and the project can be considered to be a successful one: the principles have already been taken into real-life use.

Appendix One: CV of the author

Tero Lindholm acts as a global application manager for after market business at Nokia Group. He is a finnish, married, 33 years old and M.sc in computer science. He has performed doctoral studies since 1994, being also a member a top research unit "Turku School of Computer Science" (TUCS) during 1996-1998.
Tero Lindholm's work experience includes several years of experience on managing large international projects, management consulting and quality management. He has gained experience in international line management during the last three years in his current position. Tero's special skills are project management, business process development and system development methodologies. Cross-cultural human management is one of daily activities in his current work. The common parts of process development, information systems development and human change management are the driving forces in his current work.

Figure TLINDHOL 3: Tero Lindholm

The favourite hobby is still ice hockey, where he has achieved four pronze-medals on national level. "Retired" since 1996, he is still active in sports and also an active captain of his own boat.

Contact information:
Tero Lindholm
Nokia Group
P.O.Box 86
24101 Salo
FINLAND

Email: tero.lindholm@nmp.nokia.com
Mobile phone: +358 40 545 9841.

Appendix Two: Description of Nokia

Headquartered in Finland, Nokia is a broad-scope telecommunications company supplying mobile phones, mobile and fixed telecommunications networks, data communications solutions, multimedia terminals and computer monitors. With sales in 130 countries, net sales totaled FIM 52.6 billion ($9.8 billion) in 1997. Nokia, listed on NYSE (NOK.A), employs more than 41,000 people worldwide. Operating profit exceeded 8 000 MFIM in 1997. All the key figures are rapidly growing, promising a growth of over 35 % in terms of earnings per share in 1998.

Figure Tlindhol 4: Nokia's logo.

WWW home page is at www.nokia.com.

NOKIA

CONNECTING PEOPLE

Session 2 – Metrics Driven SPI Part I

Using Defect Analysis to Initiate the Improvement Process

Otto Vinter

Brüel & Kjær, Denmark

ovinter@bk.dk

Software Metrics Applications in a European Perspective

Terttu Orci

SISU

Royal Institute of Technology

Stockholm University

Stockholm, Sweden

Using Defect Analysis to Initiate the Improvement Process

Otto Vinter
Brüel & Kjær, Denmark
ovinter@bk.dk

Introduction

This presentation will report the experiences gained from improving the software process at Brüel & Kjær. In stead of starting a process improvement programme with an assessment according to one of the common maturity models, our improvement strategy has been an experience-driven, incremental process based on the available information from earlier development projects.

We started by performing root cause analyses of error logs generated during previous development projects. Based on the findings of the analyses we then introduced improvements in our development process to prevent frequently occurring types of errors.

Once the first results from improvements had materialised, the interest in a more formal assessment could be raised. A Bootstrap assessment was performed by an external body, but because of the strong project manager culture (we are a level 2 organisation) it was not possible to turn the recommendations into improvement actions.

In order to overcome this barrier, we then involved the leading project managers through a series of interviews where they were asked about which type of problems in their development projects they felt were the most serious. The problems perceived by the project managers were quite common, so a concensus on the next improvement actions could be reached quite easily. A number of new improvement projects are now under way headed by the project managers with mentoring and support from the process improvement group.

The recommendations from the three approaches above show a remarkable overlap, so we can recommend others to use an experience-driven approach based on the available information in the company as a way to initiate a process improvement programme.

Defect Analysis

We have conducted thorough analyses and classification of bugs reported during development and after release of products. In these analyses we classified bugs according to a taxonomy described by Boris Beizer [1].

The analyses [2][3] showed the need to perform a more systematic unit test of our products. However, the analyses also showed that the major cause of bugs stemmed from requirements related issues.

The improvement actions have been funded by the Commission of the European Communities (CEC) under the ESSI programme: European System and Software Initiative. The title of the test improvement project is: PET - The Prevention of Errors through Experience-Driven Test Efforts (ESSI project no. 10438) [2][3]. The title of the requirements engineering project is: PRIDE - A Methodology for Preventing Requirements Issues from Becoming Defects (ESSI project no. 21167) [4].

The Test Improvement Project

The software quality of our company was felt to be unsatisfactory. Too many products were shipped with bugs. It was the general opinion that this was caused by a lack of testing by the developers before release.

It was therefore decided to conduct a process improvement experiment to find ways to improve the testing process. The project was titled: The Prevention of Errors through Experience-Driven Test Efforts (PET) [2][3].

The problem reports were analysed and bugs in them categorised using Boris Beizer's taxonomy [1]. We found that bugs in embedded real-time software follow the same pattern as other types of software. We found that the major cause of bugs reported (36%) are directly related to requirements, or can be derived from problems with requirements. The second largest cause of bugs (22%) stems from lack of systematic unit testing.

The techniques selected to improve unit testing were: Static and dynamic analysis. Tools were installed to support these techniques. We experienced a 46% improvement in testing efficiency (bugs found per person hour) and we raised the branch coverage of all units to above 85%.

An improved (production) version of the baseline product was then released and tracked for the same number of weeks we had measured on the existing (trial) version after its release, so that we were able to evaluate the effect of the experiment on problem reports.

The team received 75% fewer error reports than for the trial-release version of the product. Of those error reports 70% were found to be related to requirements e.g. to bugs that could not have been found through static and dynamic analysis. This once more confirmed the need for us to improve the requirements process.

In spite of these remarkable results the use of static and dynamic analysis never spread throughout the organisation. Some project managers ignored the results. Others started to work with the techniques, but stopped when time pressure built up. Those who proceeded released products with remarkably fewer bugs.

The Requirements Engineering Improvement Project

In the second improvement action we performed a closer analysis of requirements related bugs in order to find and introduce effective prevention techniques in our requirements engineering process. The project was titled: A Methodology for Preventing Requirement Issues from Becoming Defects (PRIDE) [4].

From the analysis of requirements related bugs we found that requirements issues are not what is expected from the literature. Usability issues dominate (64%). Problems with understanding and co-operating with 3rd party software packages and circumventing their errors are also very frequent (28%). Functionality issues that we (and others) originally thought were the major requirements problems only represent a smaller part (22%). Other issues account for 13%. The sum of these figures adds up to more than 100% because one bug may involve more than one issue.

This has had an impact on our methodology. It had to be focused on usability problems, and early verification and validation techniques, rather than correctness, and completeness of requirements documents.

We therefore introduced the following techniques on a real-life project:

Scenarios
Relate demands to use situations. Describe the essential tasks in each scenario.
Navigational Prototype Usability Test, Daily Tasks
Check that the users are able to use the system for daily tasks based on a navigational prototype of the user interface.

We found an overall reduction in error reports of 27%. We saw a 72% reduction in usability issues per new screen, and a 3 times increase in productivity in the design and development of the user interface.

What was also surprising was that not only did we experience a reduction in bugs related to requirements issues, we also found a reduction in other bug categories. The derived effect on other types of bugs than the requirements related can be explained by the fact that most of the developers achieved a deep understanding of the domain in which the product was going to be used from describing use situations (scenarios) and taking part in the usability tests. This invariably leads to reduced uncertainty and indecision among the developers on what features to include and how they should be implemented and work. In the previous project the new screens were constantly subject to change all through to the end of the project.

However, the impact of these techniques on the perceived quality of the released product is even greater than the prevention of bugs. Describing use situations (scenarios) enabled the team at a very early stage in the requirements engineering process to capture the most important demands seen from a user/customer perspective. The developers therefore got a very clear vision of the product before the requirements were fixed. The subsequent usability tests on very early prototypes verified that the concepts derived from the descriptions of use situations (scenarios) still matched the users' needs and could be readily understood by them in their daily use situations.

The product has now been on the market for more than 7 months and it steadily sells more than twice the number of copies than the product we have compared it to. This is in spite of the fact that it is aimed at a much smaller market niche, and that the price of the new product is much higher.

In contrast to the test techniques, the interest amoung project managers to adopt the scenario and usability techniques has been much higher. This may be because developers much rather will work with requirements than with test.

Bootstrap Assessment

When the first results of the improvement actions based on defect analysis had materialised, the management of our company could be convinced that a more formal assessment of our software development process should be performed in order to further the improvement programme.

A Bootstrap assessment was performed using the Danish company DELTA as assessors. Four projects and the software development management were interviewed by the external assessors. The overall result of the assessment was that we were at level 2.25.

The recommendations from the assessors pointed out weaknesses in the following areas:

Development Model
The shift in focus from a primarily hardware driven development to software had to be more focused. The assessors recommended to introduce a specific life-cycle for software development.

Process Descriptions
Introduce the formal as well as informal improvement actions of the software processes in the quality management system.

Unit and Integration Testing
The assessors commented that the improvements we had achieved under the test improvement experiment (PET) needed to be applied on a wider scale in the company.

Configuration Mangement
Again the assessors commented on the need for a more uniform way of performing configuration management, change control, and planning.

Requirements
The assessors were aware of the ongoing requirements engineering experiment (PRIDE) and commented on the need for a description of the process.

Project Management
Improve planning, estimating procedures, introduction of time and resource usage, closer monitoring of project progress.

The recommendations above clearly shows that a Bootstrap assessment has a much wider perspective of the software development process than defect analysis. Defect analysis primarily high-lights “hot-spots” in the development process. However, the testing and requirements “hot-spots” that we found through our defect analysis were also found through the assessment.

Due to the strong project manager culture of our company, which is typical of level 2, the recommendations from the assessment were never turned into improvement actions. Top management had stated their commitment to follow up on the assessment recommendations, but in the end they left it to the project managers themselves to find and introduce improvement actions on their individual projects. And the process improvement group did neither have the resources nor the “power” to be able to introduce new activities on a wider scale.

With no “pressure” from the top, and projects running late, it is no wonder that the project managers chose to concentrate on their day-to-day problems of getting products out of the door, so nothing happened as a result of the assessment.

Project Manager Involvement

Through the Center for Software Process Improvement, which is partly funded by the Danish state, we got the opportunity to increase our resources and research knowledge for process improvements. This became the basis for spreading improvements on a wider scale in the organisation.

We realised that if we were to succeed in introducing improvements on a wider scale, rather than those individual actions we had performed on the ESSI experiments, we had to tie the actions to the “power structure” of the organisation.

We decided to perform another evaluation of our software process, this time from the perspective of the project managers. We involved the leading project managers through a series of interviews where they were asked about which type of problems in their development projects they felt were the most serious. Seven project managers were interviewed and detailed minutes were recorded.

From these minutes we could compose a list of problem areas that the project managers found to influence their projects most. The problems perceived by the project managers correlated very well, so a concensus could be reached quite easily:

Software Development Model
The present ISO9001 registrered waterfall model was deemed unsatisfactory for efficient software development. A new model based on iterations, e.g. through experimental prototypes, was asked for. Risk management was also an important element.

Requirements
The project managers of course knew about the ongoing requirements engineering experiment (PRIDE) and requested improvements in this area.

Project Monitoring
Better estimating procedures, follow up, and progress evaluations.

Project Conclusion
Configuration management, release criteria, and testing.

Reuse
Actually this item did not initially appear in the interviews, but the organisational changes that took place at the time of the evaluation centered on establishing a group responsible for reuse.

The problems perceived by the project managers clearly resemble the recommendations from the Bootstrap assessment. Once again we see a much wider perspective of the software development process than defect analysis. However, the testing and requirements “hot-spots” that we found through our defect analysis were also found through this evaluation.

The project managers were presented with these findings at a workshop where top management also was present. They were each asked to select a topic from the list that they felt most natural to work on with their present (or up-coming project).

The process improvement group and the researchers from the Center for Software Process Improvement then established support groups that would train, mentor, and follow these projects.

Three project managers chose to work with implementing the requirements engineering techniques from PRIDE. Two chose to work with new development models. One chose to work with reuse and one with project conclusion. Finally the R&D manager wanted to contribute by improving project monitoring.

These new improvement projects are now under way headed by the project managers with mentoring and support from the process improvement group. There seems to be very little resistance to the improvement actions recommended by the support group. In fact there is great enthusiasm in the teams for both the development model experiments and requirements engineering techniques.

Comparison and Conclusions

The recommendations from the three types of approaches show a great deal of overlap though they sometimes use different words.

Recommendation	Defect Analysis	Bootstrap Assessment	Project Mgr. Interviews
Development Model			
- iterations	x	x	x
- risk management			x
Requirements	x	x	x
Project Monitoring			
- estimation		x	x
- time & resource usage		x	
- monitor progress		x	x
Project Conclusion			
- configuration mgmt.		x	x
- testing	x	x	x
- release criteria	x		x
Reuse			x
Process Descriptions		x	

Fig. OtV.1 : Comparison of Recommendations

What is important, however, is the fact that the findings from the defect analysis approach are recommended by the more general evaluations. This means that an approach based on the already available information in the company is a valid approach to initiate a process improvement programme.

We therefore conclude that bottom-up, experience-driven improvement actions based on defect analysis are a cheap an effective way to get started. Formal assessments can be postponed until the initial improvements have demonstrated the value of improving the software development process.

However, it is also clear from our experience that it is only possible to achieve a widespread acceptance of the improvement results if the process improvement group is working within, or tie their actions to, the “power structure” of the company.

Future Work

A second Bootstrap assessment is planned for January of 1999 and we look forward to seeing that the weeknesses in testing and requirements have been reduced significantly. It will probably be too early to see the effects of the other improvement actions.

Based on the new recommendations from the assessors, another round of interviews with the project managers will be performed and two more improvement actions initiated in 1999.

References

[1] Boris Beizer, *Software Testing Techniques. Second Edition,* Van Nostrand Reinhold New York, 1990.

[2] Otto Vinter, Per-Michael Poulsen, Knud Nissen, Jørn Mærsk Thomsen, *The Prevention of Errors through Experience-Driven Test Efforts. ESSI Project 10438. Final Report,* Brüel & Kjær A/S, DK-2850 Nærum, Denmark, 1996. (http://www.esi.es/ESSI/Reports/All/10438).

[3] Otto Vinter, Per-Michael Poulsen, Knud Nissen, Jørn Mærsk Thomsen, Ole Andersen, *The Prevention of Errors through Experience-Driven Test Efforts,* DLT Report D-259, DELTA, DK-2970 Horsholm, Denmark, 1996.

[4] Otto Vinter, Søren Lauesen, Jan Pries-Heje, *A Methodology for Preventing Requirements Issues from Becoming Defects. ESSI Project 21167. Final Report,* Brüel & Kjær Sound & Vibration Measurement A/S, DK-2850 Nærum, Denmark, 1998. (To appear on this web address soon: http://www.esi.es/ESSI/Reports/All/21167)

Appendix A: CV of Author

Otto Vinter is managing a software technology and process improvement group at Brüel & Kjær responsible for projects to improve the software development process.

He has been active in defining software engineering standards, procedures, and methods to be employed at Brüel & Kjær. He has also been the driving force in the company's transition from procedural programming to Object-Oriented development.

He has managed software development projects for 25+ years; with Brüel & Kjær from 1986, before that with the Danish branch of Control Data Corporation, and with Regnecentralen. He holds a Masters Degree in Computer Science from the Danish Technical University.

Appendix B: The Company

Brüel & Kjær is a leading manufacturer of high-precision measurement instruments. Brüel & Kjær develops high-precision electronic instruments for sound and vibration measurement applications. The company is headquartered in Denmark, but the majority of the products are sold through subsidiaries around the world. In the past most of the products were based on embedded real-time software, but now the number of PC applications are rapidly taking over.

Brüel & Kjær Sound & Vibration Measurement A/S
Skodsborgvej 307
DK-2850 Nærum
Denmark

Tel: +45 4580 0500
Fax: +45 4580 1405

Email: ovinter@bk.dk

Software Metrics Applications in a European Perspective

Terttu Orci

SISU

Royal Institute of Technology

Stockholm University

Stockholm, Sweden

Introduction

Software metrics has been quite an intensive research area for more than two decades, yet its practical applications have been rather limited. The 70's being the era for software size metrics, introduced lines of code, and Albrecht's Function Points [1], to measure the size and functionality of a software product. The size measurement, useful in its own right, serves also as a component in measuring productivity, and as an estimated parameter to estimation models [4],[13], which were mainly introduced in the 80's. In the 90's, the main focus is on software processes, and to improve the software processes is commonly agreed to be the solution for software companies fighting against delivery delays, cost overriding, and quality flaws in the end product. As measurement is inherent in the concept of improvement, in the stream of software process improvements currently undertaken, the number of practical applications of software metrics can be expected to increase.

A software process improvement may be undertaken with different intentions: to get certified according to a standard, e.g. ISO9001 [11], to follow the maturity ladder of the Capability Maturity Model (CMM) [12], or to solve a specific problem, e.g. the lack of configuration control.

Some of the large software companies in USA have published their long term process improvement programs including introduction of metrics, e.g. [9],[14]. We are not aware of any corresponding publications from large European companies concerning organisation wide improvement or metrics programs. Yet, in software process improvement area, several activities are in place today, the most widely known and influential being the ESSI programme (European Software and Systems Initiative) supported by CEC.

The projects within ESSI programme, usually called Process Improvement Experiments (PIEs), are normally short term, 12-18 months in duration, and intended to improve the software process, and to establish a basis for further improvement. The projects have the process improvement goal in common, but differ in the focus, and consequently in the project specific objectives. The focus may be to get increased knowledge of the state of the affairs, e.g. number of defects discovered by the customer after delivery of a product, or to investigate whether an object-oriented technology would reduce the time to market.

In the project EUREX - European Experience Exchange [5], which is an ESSI dissemination action, we have analysed the experiences reported by 13 companies, representing process improvement experiments within ESSI programme. In this paper, the experiences collected so far will be presented, with focus on how software metrics are used to measure the improvements.

EUREX

Until today, CEC has supported about 300 PIEs, some finished, others in progress. In order to disseminate the experiences and knowledge of the PIEs to a wider European audience, the project Eurex works along the objectives of

collecting,
systematizing, and
disseminating the experiences and lessons learnt

from the process improvement experiments.

The partners of EUREX are Highware Gmbh (D), Highware Productions (Fr), Gemini Societá (I), Socintecs (ES), and SISU (S). The PIEs studied have been clustered into a number of subject domains: object-oriented technology, reuse and component based development, software metrics, requirements and test, project management, and configuration management. Naturally, the domains are overlapping to a certain extent, and therefore, a PIE could, in principle, belong to several domains. For simplicity and for avoiding redundancy in the project work, each PIE is, however, placed to one and only one domain.

Each of the partners is responsible for two lines of work:

to cover the PIEs in the partner's geographic area
to make a deeper study and to merge the experiences of the PIEs in one subject

domain.

To cover the PIEs in a geographic area, a partner responsible for the area is supposed to organise one workshop in every subject domain. To such a workshop, all the local PIEs are invited to present their objectives, methods of implementation, and experiences. If a PIE cannot attend, the final or mid-term report should be made available to the workshop organising partner. The collected experiences from the workshop will then be forwarded to the partner, responsible for subject domain. That partner will finally merge and report the experiences in the subject domain in case.

SISU - the Swedish Institute of Systems Development, is responsible for the Nordic PIEs (Sweden, Denmark, Norway, and Finland) as well as for the subject domain Software Metrics. At the time of writing, two workshops have been organised by SISU. In the context of those workshops, ten PIEs have presented and/or made their final reports available. In addition to that, the final reports from three PIEs presented at a metrics workshop given by Socintecs (ES) have been made available to us. Those 13 PIEs are the basis for the study presented in this paper.

Software Metrics

Software metrics, presented in various textbooks, e.g. [6],[7],[10],[15] and conferences, 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. In the following, these aspects will be discussed in more detail.

Measurement - what is it?

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 centimeters. 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.

In the above example, the representation condition is easy to understand and accept. The difficulties arise as soon as we are talking about metrics which have been defined although there does not exist such an intuitive understanding of the property. Good candidates in this class are complexity of code and quality of a product. The complexity is often measured by McCabe's cyclomatic number, while the most used quality metrics are related to the defects: the number of defects/KLOC, or the number of defects discovered by the customer. None of those metrics paint the whole picture, mainly because there does not exist that intuitive understanding in the first place. Both metrics give only a partial view of the property of interest.

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.

A measurement on a *nominal* scale is a classification of the entities into a set of disjoint classes without any ordering between the classes, i.e. the classes form a partition. An example of measurement on a nominal scale is a categorization of defects, if no ordering of the defects is of interest. The *ordinal* scale extends the nominal by adding an order for the classes. An example of the ordinal scale is categorizing defects with respect to severity. The *interval* scale adds the concept of distance, and poses the requirement that the distance between any two consequtive classes is to be equal. Celsius and Fahrenheit for measuring temperature are on interval scale. *Ratio* scale is the interval scale extended with zero-element, i.e. it includes the total absence of the property in case, e.g. size of code measured by LOC, which in principle could be zero. An *absolute* scale measurement is counting the occurrences of entities, e.g. hours spent on a particular task.

It is important to establish the scale of the measurement in that different scales allow different manipulations with the measurement data. On the nominal scale, only the *frequency* and *mode* are relevant. On ordinal scale we can calculate the *median* of the values. On interval scale also the *arithmetic mean* is meaningful to calculate, while on ratio scale, also *geometric mean* makes sense. On absolute scale, we can do all the calculations by numbers.

If a measure has components from different scales, the scale with the weakest manipulation possibilities determines the scale of the composite measure. For example, productivity calculated by dividing the size of the output by effort needed to produce that output is on ratio scale because size is on ratio scale, although effort is on absolute scale.

Attributes can be classified as *internal* and *external* attributes. Measuring internal attributes of an entity, no other entities or attributes are involved. An example of an internal attribute is size. Measuring an external attribute of an entity, other entities or attributes may be needed for measurement. An example of an external attribute of the entity code is maintainability. To measure maintainability, it is not enough to study the code, but the code in the process of error detection and correction must be studied. The external attributes are the challenging ones when it comes to define and validate metrics. If there is no adequate metric for an external attribute, e.g. maintainability, it may be possible to predict from an internal attribute. This is what we always do when buying a car: we cannot measure the external attributes like how well the car behaves under certain circumstances, but use internal attributes like tire dimension, brake system, or whatever, as a predictor.

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.

Why is the triangle - the entity, attribute, and metric - needed?

Unless there is a clear statement of the entity, attribute, and a 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.

What types of use are there for the measurement data?

The measurement data has two principal types of use: *assessment*, i.e. to understand the state of the affairs, and *prediction*, i.e. to make a statement about the future state of the affairs. Possible uses of measurement data in prediction are as *input* to a prediction model, to *calibrate* a prediction model, or to serve as the basis for *defining* a prediction model. An example of an assessment is measuring complexity of code by McCabe’s cyclomatic number. An example of use of McCabe's cyclomatic number is as input to a prediction model to predicting the attribute maintainability of the code. There are studies showing that McCabe's metric is not a valid predictor of maintainability, nor is it an acceptable complexity metric [6]. Yet it is often used in both senses.

Validating metrics and prediction systems?

The software metrics literature offers a large number of metrics, yet there are severe limitations in terms of adequate metrics missing, e.g. metrics for complexity and quality of different kind of products, size and structure of products other than code, to take a few examples. If there does not exist a metric adequate to the purpose, an organisation can always define its own metrics. When defining metrics, empirical data is needed to validate the metric in case. The validation is intended to assure that the representation condition is fulfilled, i.e. that the empirical relations are preserved and preserve the numerical relations. First of all, there must exist consensus, an intuitive understanding of the attribute. The basic idea of the metrics validation is that it should measure the attribute it is intended to measure. A good example of an attribute, the metric of which is often said to fail in this respect, is the IQ test. It is claimed that the test does not measure the attribute intelligence, but something else, for example logical ability. Logical ability, although possibly being a component of intelligence, is not the lone carrier of intelligence. The problem with the metric for intelligence is that there is no commonly agreed definition of intelligence in the first place.

The validation requirement applies also to prediction systems. A prediction system is valid if it correctly predicts what it is intended to predict. In validation of a prediction system, the empirical data must cover the predictive capability of the prediction system, i.e. to give a measure how much the estimates differ from the actuals. Metrics for this purpose are MRE (Mean Relative Error) and PRED (Prediction accuracy).

How to identify right metrics?

It is stressed by most authors that the identification of metrics should be goal oriented, i.e. unless you know the purpose of the measurement, you should not start a metrics programme. The most well-known methods for identifying the right metrics from the goals are the Goal-Questions-Metrics method (GQM) [3] and Application of

Metrics in Industry (AMI) [2]. Although the methods are not difficult as such, the difficulty lies in inventing the right questions and metrics, which is not trivial at all. Knowledge and training in software metrics as well as familiarity with the application domain are needed.

What is good data?

The measurement data collected should fulfil certain quality criteria to be useful in an analysis. We do not attempt to give an exhaustive listi of quality criteria, rather to present some requirements which are commonly accepted as necessary requirements on measurement data, namely

correctness
consistency
time precision
right granularity

The *correctness* requirement means that the data should be collected according to the rules of metrics definition. For example, if measuring the code size by LOC, and the rules state that reused code should be excluded, correct data would not include reused lines of code. Needless to say, this requirement implies that there should be precise definitions in the first place. The *consistency* criteria means that different people measuring a product for some purpose should use the same version of the product, i.e. the product to be measured should be under configuration control. The *time precision* means that if a process is measured, there should be distinct start and end of the process, otherwise the process measurement is meaningless. The *right granularity* means that the granularity should be determined with respect to the goals of the measurement. If the granularity is too fine for the purpose, unnecessary effort must be put into data collection, while in the opposite case, the usefulness of the data might be reduced.

How to introduce a metrics programme and to keep it alive?

It is often argued that only simple metrics should be used, whatever is meant by simple metrics, and that metrics should be automatically collected, otherwise the programme will fail in the long run. The first question to ask is how long the program should run? Has the programme been introduced with the purpose of assessing the state of the affairs, to solve a particular problem, or has the measurement programme been included in a continuous process improvement programme, intended to run long term. There are a number of guidelines and good advice for introducing a metrics programme and to keep it alive [6], [14], e.g.

get and keep a commitment from the top management
measure anything, but individuals
give feedback to those who collect the measurement data
metrics should be automatically collected if possible
don't measure unless there is clear purpose with it

The guidelines do not, of course, guarantee a success of a metrics programme, but the lack of the requirements may easily lead to a failure. Of particular importance is the commitment from the top management. Without that support, the programme has not much chance to survive.

The Study

In this study, 13 PIEs have been analyzed. The basis for the structure of the analysis has been the theoretical aspects and the questions of practical interest presented in the previous section. In particular, we analyse the PIEs with respect to the following questions:

Is the measurement well-defined?
Have the metrics and prediction systems been validated?
What is the distribution of the entity types product, process, and resource?
Has expert support been needed in metrics definition?
How have the right metrics been identified?
What are the lessons learnt and problems encountered?

Is the measurement well-defined?

With this question we try to understand the metrics maturity of the PIEs, which can be said to represent the European software community. There are several requirements on being well-defined. For example, if LOC is used to measure the size of code, the model of the entity should be precisely defined, e.g. will comments be included or not. In this study, we only ask whether or not each of entity, attribute, and metric have been defined in the context of a metric presented. If a concept is not explicitly stated, but obvious from the context, it is regarded as defined.

In the 13 PIEs analysed, 51 different metrics were presented in total. Only nine of the metrics definitions (18%) included the *entity, attribute* and *metric*, while in the remaining cases, one or two of the concepts were not explicitly stated, nor clear from the context.

In 27 of the metrics (53%), the *entity was not defined*. For example, the metric “number of errors discovered within three months after release” might be intended to measure a process, e.g. development or testing. Alternatively, it might be intended to give an indication of the error density of the products delivered to a customer. To take another example, the metric “percentage of the problems solved at the first level”, is stated to measure the attribute quality, but it is not clear which entity it refers to. In principle, it could be a product or a process.

In 28 of the 51 metrics (55%), the *attribute was not defined,* in some of them with the entity defined, in some without. An example of a *missing attribute with defined entity* is the metrics “the number of nodes in the design tree” characterizing the entity design document. The attribute intended might be the size of the design document. This metric could also be used to predict some attribute for another entity, e.g. maintainability of the code. Another example of a missing attribute with a defined entity is "the number of software problem reports" for a product. This attribute is often used to assess the quality of a product. In that case, the number of software problem reports should rather be related to the size of the product, instead of giving it as an absolute number. An example of both *entity and attribute missing* is the metric "the number of software errors per function point". It might been used to assess the attribute "error density" of a product, or it could be a metric for assessing the quality of the development process.

In five of the 51 metrics presented (10%), the *metric was not defined*. An example of a missing metric with other concepts in place is "risk" assessment for a process. An example of both metric and entity missing, is the attribute "productivity", usually a resource metric, but without explicit definition, we do not know what kind of resource is intended, neither how it is measured.

In 20 of the 51 metrics presented (39*%), neither the entity nor the attribute were defined*. Examples of such metrics are "percent of cost per function point", "the number of claims", "the number of problems registered", "the number of test cases", and "the number of unnecessary modifications related to the total number of modifications".

Have the metrics and prediction systems been validated?

Unless the metrics and prediction systems have been validated, the value of the measurement data is not of much interest. Even worse, it may be misleading in that the measurement data can be interpreted in several ways.

Only two of the 13 PIEs (4%) have presented experiments for validation, both validating prediction systems. Two of the PIEs have used an invalid metric, McCabe cyclomatic number to measure the complexity of the code. With a closer look, it seems that the real use of McCabe metric has in those PIEs been to predict the maintainability of the code, the higher the McCabe number, the more efforts can be expected to be needed in maintenance. However, McCabe metric is neither a valid predictor of maintainability [6].

What is the distribution of the entity types product, process, and resource?

The PIEs are supposed to measure the improvement obtained because of the process improvement experiment. It could be expected that process measurement would be the main entity type in the measurements. However, of the 24 metrics with entity defined at all, the distribution of the entity types is as follows: eight process metrics (33%), 13 product metrics (55%), one resource metric (4%), and two global metrics (8%) measuring the organisation maturity. It seems that product metrics are to a large extent used to measure some external attribute of a process, e.g. the product quality measured by the number of problems can be used to measure the quality of the software process, or some of its activities.

Has expert support been needed in metrics definition?

A need of expert support in defining metrics would imply some lack knowledge and training in software metrics in the organisations.

Of the 13 PIEs studied, seven of them (54%) state explicitly having used expert support to define the metrics. This number could be higher in reality, as there is no reason for the PIEs to present this fact unless the metrics expert was a partner in the project.

How have the right metrics been identified?

With this question we intend to capture the methodologies used to identify right metrics.

The methods for identifying the right metrics have been Goal-Question-Metric (GQM) in one case (8%), and AMI (Application of Metrics in Industry) in one case (8%). As 46% of the PIEs had expert support in metrics definition, the explicit method of identifying the metrics might not be known to the PIE, and therefore not mentioned.

What are the lessons learnt and problems encountered?

The reason for this question is to investigate whether the lessons learnt and problems encountered confirm or conflict the common knowledge of introducing metrics programs. A further reason is to use the problem statements to support the discussion and conclusions of the study. We state all the problem and experience types explicitly and once. Most of them are shared by several PIEs.

Data was not consistently collected
The purpose of measurement was not clear
Reliability and accuracy of the measurement data is important
There was resistance by people to collect metrics data
Collection of data must be integrated in working processes
Metrics program was too ambitious

It was difficult to measure
Metrics must be defined and validated
Relationships between metrics must be defined
There must be templates for metrics data collection
Collection of data was a big effort
It was useful to get the metrics
Use simple metrics

None of the statements is new, nor in conflict with the common knowledge in the area. It is worth to observe that the PIEs stress the requirements on correctness, reliability, accuracy, consistency, definition, and validation, especially as the metrics presented in large extent lack these properties. The need for templates for data collection can also be interpreted as supporting the need of precise definition of data. The statement of ideally using simple metrics may be interpreted as the defined metrics have been difficult to understand, the metrics data collection has been difficult, or the analysis part has been difficult. Similarly, the difficulty in measuring may be based on difficulty of understanding because of lacking precise definitions. It is well-known that resistance for metrics data collection and filling in the forms is common, and that it may be, to some extent, be reduced by giving feedback of the measurement results.

Conclusions

We have analysed 13 PIEs with the focus on how the improvement has been measured in the process improvement experiments undertaken. The basis for the analysis consists of a number of aspects, originating from both theory and practice.

The aspect which we believe is of most importance, is the definition and validation of the metrics used. Although this issue has its origin in theory, it is not only an academic issue to be discussed between software metrics researchers, but it is an essential issue for the practical applications. Without a well-defined and validated metrics and prediction systems, measurement has no value to the organisation. Even worse, the measurement data may be misleading as without precise definitions, all interpretations become possible. Without precise and validated metrics, comparing measurement data from different organisations does not make sense either.

From this point of view, the PIEs under study appear considerably 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. Still, it seems correct to draw the conclusion that there is major potential for improvement in the software metrics maturity and understanding the importance of the underlying theory. The conclusion is also supported by the lessons learnt, presented in the previous section: Without a precise metrics definitions it is not surprising at all that it has been difficult to measure, and that the reliability, accuracy, correctness, consistency, definition, and validation aspects have been included in the lessons learnt. Another supporting fact is that more than half of the PIE organisations needed expert support in metrics definition. Yet, the attributes measured were, with a few exceptions, internal attributes, the metrics of which are easier to define and validate than metrics for external attributes.

To obtain an improvement in the software metrics maturity on a European level, software engineering research with the focus on empirical studies, especially in defining and validating metrics, is needed. The empirical data needed for the research should ideally originate from the industry. A cooperation between research and industry is needed to getting started in a larger scale towards a software metrics maturity improvement. To ensure a continuous improvement in metrics maturity, software metrics should be included in the software engineering curricula, to train the top management and engineers of tomorrow.

References

[1] Albrecht, A.J., Measuring Application Development, in *Proceedings of IBM Applications Development Joint SHARE/GUIDE Symposium*, Monterey, CA, pp. 83-92, 1979.

[2] AMI – Application of Metrics in Industry, ESPRIT metrics technology transfer project 1991-1993.

[3] Basili, V.R., Rombach, H.D., The TAME project: Towards improvement-oriented software environments, in *IEEE Transactions on Software Engineering* 14(6), pp.758-773, 1988.

[4] Boehm, B.W., Software Engineering Economics, Prentice Hall, 1981.

[5] EUREX – European Experience Exchange, Project Number 24478, ESSI Dissemination Action, Annex I, Project Programme.

[6] Fenton, N.E., Pfleeger, S.L., Software Metrics – A Rigorous & Practical Approach, International Thomsom Publishing Inc., 1996.

[7] Fenton, N. Whitty, R., Iizuka, Y., Software Quality – Assurance and Measurement. A Worldwide Perspective. International Thomson Computer Press, 1995.

[8] Gillies, A.C., Software Quality – Theory and Management, 2nd ed, International Thomson Computer Press, 1997.

[9] Grady, R.B., Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, 1992.

[10] Kan, S.H., Metrics and Models in Software Quality Engineering. Addison-Wesley, 1995.

[11] Oskarsson, Ö., Glass, R.L., ISO9000 i programutveckling – att konstruera kvalitetsprodukter, Studentlitteratur, 1995.

[12] Paulk, M.C. et al, The Capability Maturity Model – Guidelines for Improving

the Software Process, Addison-Wesley, 1995.

[13] Putnam, L.H., Fitzsimmons, A., Estimating software costs, in *Datamation*, Deptember, October and November 1979, pp. 312-315.

[14] SEI, Managing Software Development with Metrics, Course material, 1996.

[15] Shepperd, M. Foundations of Software Measurement. Prentice Hall, 1995.

Session 3 – Implementation of SPI Part I

Data Driven Improvement for SMEs

Tor Stålhane, Ph.D.,
Kari Juul Wedde, MSc,
Tore Dybå, MSc.
SINTEF, Trondheim, Norway

SPI in Embedded Software Applications

Bjarne Månsson
Software Group Manager
BARCO Communication Systems Denmark

An Experience of SEPG Organization

Alessia Billi
Sodalia S.p.A., Trento- Italy
alessia@sodalia.it

Data Driven Improvement for SMEs

Tor Stålhane, Ph.D.,
Kari Juul Wedde, MSc,
Tore Dybå, MSc.
SINTEF, Trondheim, Norway

Introduction

As the competition in the software development business grow fiercer, process improvement becomes more and more important. The best way for a company to improve is through learning from their own data and experiences. However, experiences are basically individual and in order to move from the individual experiences of each developer to something that is useful in the company at large, we need a method for converting data and experience into reusable knowledge through data analyses and interpretation. This way of improving is called data driven improvement.

In order to perform data analyses, it is necessary to combine collected data with experiences that are available in the organization. This is important in order to decide:

- What to measure and how to measure it.
- How to make sense of the collected data.
- How to move from experience and collected data to reusable knowledge.

In statistics, the problem of combining data and experience is usually solved within a Bayesian framework. However, in order not to relay heavily on the formalization of the prior – expert – knowledge, we needed a more informal approach. An example of such a way to combine data, prior knowledge and experience is the feedback sessions used in GQM [3].

It is outside the scope of this paper to report on our experiences with extending the knowledge created inside a team across to other teams and to the organization as a whole. Besides the need for building up an experience bases, an additional - but often ignored role of data collection - is to dispel myths, which act as roadblocks for the improvement work. Such myths are often unconscious acts of self-defense in the company. As long as the myths are not challenged, they block all thoughts of improvement. Thus, only by dispelling the myths can we move on to changes and true process improvement.

The rest of this paper is organized as follows: First we discuss learning in organizations in general and how this relates to the special problems facing SMEs. Then we will go into more details and focus on how SINTEF has solved the identified problems by using two approaches, namely GQM and risk based improvement. At last we will sum up our experiences and offer a set of conclusions.

On Organizational Learning

The need for learning

Process improvement based on "best practice" models (e.g. CMM and Bootstrap) and SPC can be contrasted with the use of improvement processes that are mainly concerned with the contingent and human characteristics of software development. In our view, software development is a social process where the resulting software cannot be separated from the actors engaged in developing it. This perspective requires *learning* rather than introduction of "best practice" models or SPC to accomplish improvements in software development processes.

Fiol and Lyles [7] suggest that organizational learning is "the process of improving actions through better knowledge and understanding." We agree with this viewpoint, and hold that the role of organizational learning, within the context of software process improvement, is to provide *a framework for improved actions*. However, in order to understand how such a framework could be used, there are *two basic dimension of software process improvement* that must be conceived. One has to do with the *type of situation*; the other has to do with the *type of learning*. For software organizations in general, it is important to be alert to the fact that some combinations of these dimensions facilitate improvements, while other combinations inhibit improvements. This is summarized in Table 1.

Type of Situation	**Type of Learning**	
	Single-Loop	Double-Loop
Stable (standardized)	Facilitates Improvement	Inhibits Improvement
Turbulent (innovative)	Inhibits Improvement	Facilitates Improvement

Table 1. The two dimensions of software process improvement.

The *situation* in software development processes is a sequence of stable and turbulent conditions demanding alternations between innovation and standardization. This suggests that process improvement require both change and stability. Fiol and Lyles elaborate on this, noting that too much stability within an organization can be dysfunctional and that too much change and turbulence leads to difficulties for people to make sense of their environments. In other words, software process improvement involves the creation and manipulation of this tension between constancy and change.

The *learning dimension* consists of what Argyris and Schön [1] call "single-loop" and "double-loop" learning. They define single-loop learning as "instrumental learning that changes strategies for action or assumptions underlying strategies in ways that leave the values of a theory of action unchanged." Double-loop learning, on the other hand, is defined as "learning that results in a change in the values of theory-in-use, as well as in its strategies and assumptions."

The concepts of single-loop and double-loop learning are crucial in understanding the restructuring of the software organizations' routines and practices. At their best, SPC-based improvement models are concerned with how to achieve better effectiveness within the existing values and norms, that is, single-loop learning. Often, however, they are connected with simple adaptation rather than learning. Furthermore, they are concerned with solving the needs of large organizations, operating in highly stable environments with long-term contracts. This can be contrasted with the majority of SMEs that operate in increasingly changing environments where the periods of stabilization are constantly shortened, thus requiring adeptness to double-loop learning and reflective practice.

In sum, both standardization and innovation can produce improved actions for SMEs in some situations, but can also harm these organizations in other situations. Consequently, only by recognizing this challenge of the "learning paradox", and the intrinsic short periods of stabilization facing most SMEs, can they expect to succeed with software process improvement.

Creating knowledge from experiences

The most powerful learning comes from direct experience. However, to understand how SMEs can learn from such experiences, we must first understand the nature and forms of humane knowledge and the processes whereby this knowledge is created. Kolb [9] defines individual learning as "the process whereby knowledge is created through the transformation of experience". This experiential learning process, builds on Lewin's [11] model of action research, and consists of four stages in a cycle. First we have the concrete experience. This is followed by collection of data and reflective observation. Next comes abstract conceptualization, where models are constructed to define, explain and possibly predict what we observe. Finally, in active experimentation, models and ideas are tested in new situations. The outcome of the experiment becomes concrete experience.

There are two aspects of the experiential learning model that are important for process improvement in SMEs. First, the emphasis on here-and-now, concrete experience to validate and test abstract concepts. Second, the concept of feedback to describe a social learning and problem-solving process that creates knowledge. This feedback loop provides the basis for data analyses and goal-oriented action.

Whereas Kolb's theory is individually oriented, Nonaka and Takeuchi [12] have presented a theory of knowledge creation that is team and organization oriented, emphasizing Polanyi's [16] distinction between tacit and explicit knowledge. Tacit knowledge is personal and context specific, and therefore hard to formalize and communicate. Explicit knowledge, on the other hand, is transmittable in formal, systematic language. Furthermore, human beings acquire knowledge by actively creating and organizing their own experiences and only a part of this knowledge can be expressed in words and numbers (Polyani).

Nonaka and Takeuchi define organizational knowledge creation as a continuous and dynamic interaction between tacit and explicit knowledge. This interaction consists of four modes of “knowledge conversion”, as shown in Figure 1. First, the socialization mode starts by building a “field” of interactions, letting the members share experiences and creating tacit knowledge. Second, the externalization mode helps team members to engage in a process of converting tacit knowledge into explicit concepts. Third, the combination mode lets organizational members systematize and share newly created explicit concepts, and existing knowledge into a knowledge system. Finally, internalization or “learning-by-doing” embodies explicit knowledge into tacit knowledge.

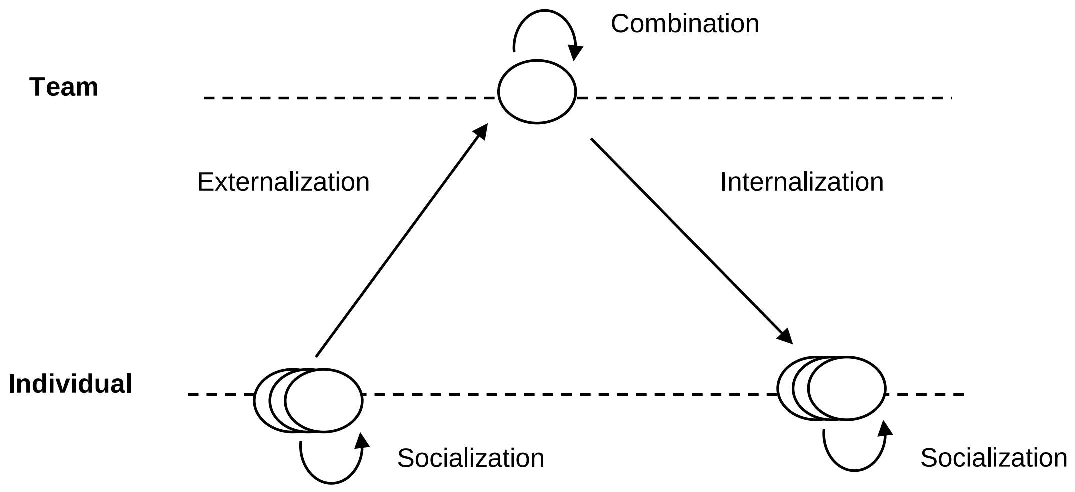


Figure 1. Interaction between tacit and explicit knowledge at the team level.

Externalization holds the key to organizational knowledge creation, since this creates new, explicit concepts from tacit knowledge [12]. *Unless shared knowledge becomes explicit, it cannot easily be reused*. The organization cannot create knowledge on its own without individual initiative and interaction at the group level - teams play a central role in the knowledge creation process in SMEs since they provide a shared context in which individual developers can interact with each other. Consequently, we have primarily focused on software development teams.

For learning to become more than a team level affair, however, knowledge must be spread quickly and efficiently throughout the whole organization. One powerful method of such diffusion is through the use of computer-based organizational memory (Huber, [8]) or an experience base (Basili, [4]). This topic will, however, not be discussed any further in this paper

The Challenge for SMEs

In the face of the need for learning in order to improve, and that we need to base this learning on our own experiences, the SMEs face two challenges, namely an ever-changing environment - only partly controlled by themselves - and few projects running at any given point in time. As a result of this, they have few data, which they can analyze and use to build up an experience database. In addition, collected data will soon be outdated and left irrelevant or – in the best case – uncertain.

These things taken together implies that SMEs need to grab the data as soon as they are available, extract the important information for learning and convert it to improvement actions without collecting long time series or amass large amounts of data needed for a tradition statistical improvement approach.

When we have collected the data according to the GQM plan and performed the necessary analysis, we can select one of two approaches, depending on the kind of approach we have chosen:

- We can create knowledge that can be used for later improvement.
- We can use the results directly to search for improvement opportunities.

Irrespective of which approach we chose, we will get a lot of small but important improvement steps from the developers during the measurement and analyses processes.

In the next section, we will first describe our overall approach to process improvement within the GQM framework. We will then go on to discuss the two selected improvement approaches and how they have been used in a large national project called SPIQ.

The SPIQ Approach

The SPIQ (Software Process Improvement for better Quality) is a major Norwegian software improvement program. The overall goal of SPIQ is to increase the competitiveness and profitability of Norwegian IT-industry through systematic and continuos process improvement.

SPIQ is based on the general process improvement principles of Total Quality Management (TQM) [2] [19], [20]. Specifically, the Plan-Do-Check-Act (PDCA) cycle is important. Figure 2 shows the SPIQ improvement process - a two level PDCA cycle, the project level and the organization level.

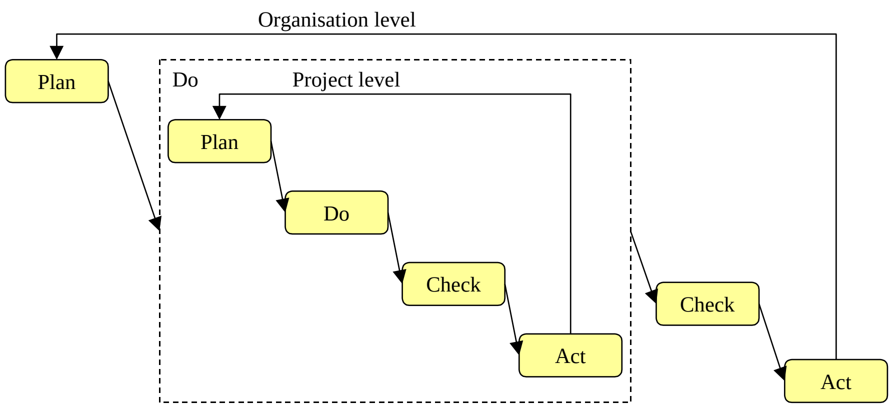


Figure 2 Two level PDCA cycle

The inner loop of this model is realized by one or more Process Improvement Experiments - PIEs. The PIEs are implemented according to the ESSI model where improvement project and development project are managed as two separate projects – but with strong relations. The development project is a real project; not an experiment set up just for the PIE.

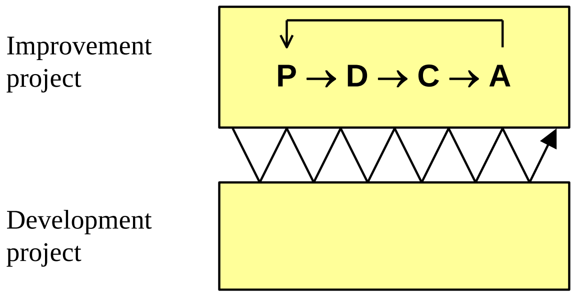


Figure 3 Process Improvement Experiment

When we implement the PDCA cycle on the project level, GQM [14], [15] is our most important tool for the Plan and Do steps. The fundamentals of GQM are the Goals, Questions and Metrics, documented in a GQM plan. The Plan part of PDCA covers the definition of Goals, Questions and Metrics and the Do part is the implementation of the GQM plan - including feedback sessions. The Check part is covered by the Post Mortem Analyses - which may be seen as a special feedback session - and the Act part consists of feeding the experiences back into the organizational level for institutionalization. For the institutionalization we are using the principles from Experience Factory [5].

The conclusion from an earlier ESPRIT project is that feedback sessions are the single most important element in keeping a measurement program alive. This is consistent with our experience. Feedback sessions are important both as a means to motivate the project team and as a method for data analyses and learning.

Another element that is important is the use of group interviews to define the GQM plan – i.e. to define what to measure and how to measure it. Group processes – including interviews and feedback sessions – are used to move from data and individual experiences to **shared explicit knowledge**.

In addition, we have good experience with using TQM tools – simple tools like histograms and scatter plots – in combination with GQM's feedback sessions to analyze data. TQM tools are important in order to communicate and to make sense of the collected data.

Learning from Experience in SMEs

SINTEF Telecom and Informatics has worked with SMEs and improvement projects for years. Together with Norwegian industry we have participated in ESSI PIEs and improvement projects run under the Norwegian software improvement program SPIQ. This chapter describes our experience from these projects with respect to the problems identified in the introduction chapter.

This includes our experiences in going from individual tacit knowledge to shared explicit knowledge that can be reused by software development teams. We discuss the importance of involving the whole team and our experiences in using group interviews and feedback session at the team level to interpret data and to share experiences. Finally, the importance of writing experience reports from the PIEs will be exemplified, as an important action when converting tacit knowledge into explicit knowledge.

Involve the whole project team

"Why should *I* collect data?" A developer in a company asked this when they tried to implement a measurement program the old way, i.e. by just deciding to collect some metrics and then ask the developers to provide the data.

It is our experience that it works much better if the company involve the whole project team from the start, i.e. from the moment the development project for a SPI project is chosen. The overall improvement goal will normally be known by then. The same goes for the GQM Goal.

In GQM the people representing the Viewpoint of the GQM Goal are considered to be the experts. In the context of process improvement the project team always represents the Viewpoint and as such they are the experts. They should therefore be given a chance to confirm whether the defined SPI project is relevant for their project or not. Further more they should take part in defining Questions and Metrics related to the Goal.

It is often claimed that software developers are not interested in SPIs, they are technology driven. Technology is important for the software community and technology is therefore of great importance for the developers. Our experience, however, is that they also take interest in SPIs if they are properly informed and allowed to participate from the beginning. In SPIs performed the GQM way, the process improvement is bottom up as soon as the Goal is given. The bottom up approach helps us to collect metrics and solve problems that the developers consider important.

Group interviews

In SPIQ we are using group interviews involving the whole project team in order to define the Questions and Metrics – i.e. to fill in the GQM abstraction sheet. This is possible in SMEs where few project teams have more than 10-12 members. In the description of the GQM process this step is always conducted by interviewing each team member separately. Individual interviews may be necessary for non-homogenous groups, in order to prevent any single member from dominating the process. We used individual interviews in one project, but found it both time consuming and difficult. Several iterations were needed before we had a plan the members could agree on, and the result was not any better than the ones we obtained by using the group process.

By using a group process, the team can discuss and obtain an agreement during the meeting. To overcome the problem with non-homogenous teams and dominating persons, we start the session by splitting the team into groups of two persons. In a group of two persons nobody could just drop out and all team members had a fair chance to come forward with their own ideas. These two-person groups were given 15 minutes to come up with a set of Questions. The questions from all groups were written on a whiteboard and served as a starting point for further discussion. The whole session takes two to three hours and the result will be a draft GQM abstraction sheet.

The GQM abstraction sheet is a means for acquiring the information needed for defining the GQM measurement plan. The abstraction sheet has four quadrants – see Figure 4. The upper quadrants show the Questions while the lower half shows the hypotheses. The hypotheses are important in order to verify the validity of the Questions. If the team have difficulties in coming up with any hypotheses, they may have defined the wrong Questions. The hypotheses are, however, often dropped. This is especially true for the Baseline Hypotheses.

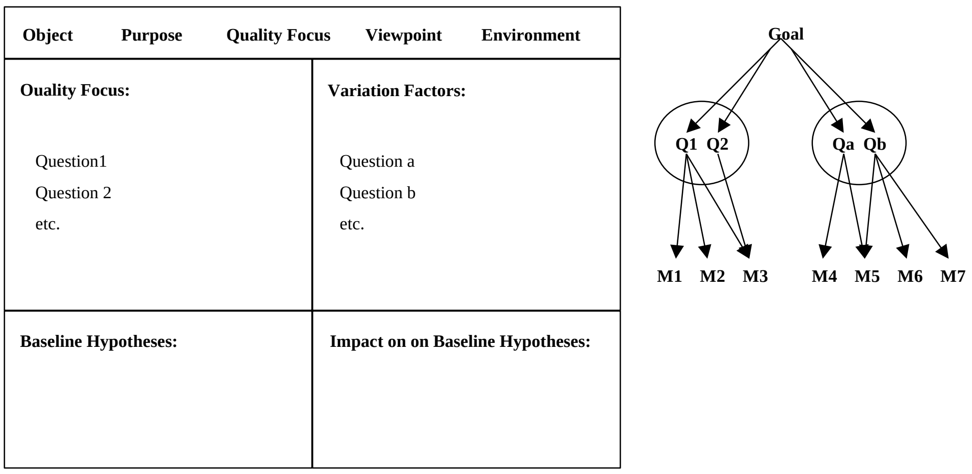


Figure 4 GQM abstraction sheet

The Baseline Hypotheses quadrant shall document the developers' view of the current status of the measured properties. This means that the team should use their current knowledge about the process to answer the defined Questions. In one company they did not see any reason why they should fill in this quadrant, but they started the job anyhow. Filling in this quadrant, however, started an enthusiastic discussion, involving all team members. The discussion served as a great motivation factor for the measurement program and when the data collection started, they were all eager to see the result of the measurements.

All together, group interviews is an efficient way to fill in the GQM abstraction sheet, giving a large range of important effects:

- Individual tacit knowledge is converted into explicit shared knowledge
- Developers are motivated to participate in the measurement program
- We get a measurement plan that is based on the organizations overall objectives and the developers immediate problems

Data analyses in the GQM context

When GQM and QIP were published, several of its proponents suggested using the collected data to repeat successful projects through an SPC approach. This is, however, impractical. In [13] Ould has given some reasons for this. We have two additional reasons:

- We do not usually have a deep understanding of the reasons why a project was a success. Thus, there is a strong possibility that we focus on the wrong factors, for instance tries to follow the same distribution of resources spent per phase, when the real – and may be only – reason for the success was that the project was staffed with highly experienced people.
- Each process improvement action will change one or more aspects of the process. As long as we do not have complete understanding of all factors that are involved in the outcome of a project, we can not update our old data (previous to the change) in order to make them relevant for the changed process.

Our solution to this is to use expert opinion aided by simple plots to interpret the data. In addition we have included simple statistical techniques. Our data analysis approach is as follows:

1. Select an appropriate plotting technique. The SPIQ project has provided guidelines for this, depending on data type and amount.[10] In order to get a free discussion, it is important not to add any analyses at this point.
2. Use the feedback sessions to present the data to the developers. There are several possible responses to the presented data, each giving rise to a different action:

- The developers do not believe the data – “This is impossible!” In this case, no further analysis is useful. We need to find out why the data are not considered reliable and correct the data collection accordingly. This does not necessarily lead to new data being collected. The solution may also be to collect the data in a way that convince the developers that the values are indeed correct.
- The developers recognize that the data collected are incomplete - “We need other data sets in order to make sense of this”. A typical example is failure data where we have mixed data from new components and reused components or from simple and highly complex components. The solution is to improve the data collection process to cater to the differences in data.
- The developers are able to interpret the data directly from the plot - “As

we thought. This is because so and so but the manager will not listen!" In these cases we have gained new or validated already available knowledge about the process. Such knowledge can later be used to pinpoint improvement opportunities. A typical example is that the developers get confirmation on that more time spent on preparation before a review leads to more bugs found.

A statistical analysis is only necessary in the third case – where the developers are able to provide a direct interpretation of the data plot. The question posed to the statistician in this case is "How likely is it that the observed pattern is not the result of some random effect?" The real challenge for the feedback session comes when the conclusion is that the level of significance is, say, 30%.

Note that it is dangerous to jump for instance from "A is linearly dependent on B" to "B causes A". It is always a possibility that both A and B depend on some unobserved variable C. If C is not observed, "B causes A" and "C causes A and B" will give the same kind of observations. In order to sort this out, it is important to use the knowledge and experience of the developers during data analyses.

Feedback sessions

Feedback sessions are well organized, structured meetings integrating the project team and the measurement team. In these meetings the collected data are presented to and interpreted by the project team members. Combining the knowledge of the team members with available data, we will get the following results:

- Identification of immediate changes and modifications for both the software development process and the measurement process
- Identification of long term improvement opportunities

The results of the discussions have to be summed up in feedback session reports that will be input to the post-mortem analyses at the end of the project. Writing these reports is an important act of converting tacit knowledge into explicit knowledge for the company.

Seen from a learning perspective, the feedback sessions thus gives results on many levels. The project team will learn as individuals and as a group and the organization collects experiences that can be useful in other projects later on. All together the result can be summed up as:

- Individual experiences for the team members
- Measurements give knowledge of how things really are
- Common understanding of what we can learn from this knowledge
- Explicit shared knowledge about the process that can be reused by the team and fed back into the organization level

The role of the TQM tools

Communication is important for learning, and to communicate we need means that can be understood by all. The TQM tools are well suited. Most of them – like histograms, scatter plots etc, are well known and therefore help to ease communication in the whole organization. In addition to being suited for communication inside the project team, they are well suited for communicating the results of the improvement project to the management. For an excellent summary see [18].

Risk in Data Driven Improvement

If we want to move on from data collection to improvement we have to deal with the uncertainties inherit in an approach based on a small data set and expert judgement. These uncertainties can be handled through risk driven improvement activities. The rather short time available is an extra complicating factor in this picture. Thus, risk rises from two aspects of process improvement:

1. Few data collected over a short time span, which implies an uncertain diagnostic of the process
2. Uncertainty in understanding, which leads to uncertainties regarding the effect of the proposed improvement actions.

As mentioned before, SMEs have few projects going on in parallel – usually only one or two each year. The success of these projects is a critical factor in the survival of the SMEs. Thus, the experiment performed in order to improve needs to be well controlled so that the chosen approach can be changed immediately if things seem to go wrong. These problem factors imply that we must be careful when choosing an improvement strategy. In our opinion two things are critical:

1. Use many small improvement steps, not a few giant leaps. Evolution, not revolution.
2. Design an improvement process with short feedback loops. This will give us a better control and quicker reactions in case of problems.

Our experience has shown that data collection in most cases is more efficient for identifying problem areas than for problem removal. Problem removal depends critically on the knowledge, experience and motivation of the developers. In order to tap this reservoir of information, SINTEF uses root cause analyses [18], realized by a two step approach, starting with Pareto analyses in order to identify the most important problems. We then use a group process, supported by an Ishikawa diagram to analyze problem causes. When we feel confident that the right root causes have been identified, we start to work on the improvement steps. This last step is the easiest one. In fact, one company has stated that when the Ishikawa diagram is completed, the solutions are in most cases obvious.

Below is an example of an Ishikawa diagram. The main cause – Incomplete requirements specifications – has been identified as an important problem cause by collecting and analyzing data in a feedback session, supported by a Pareto analyses.

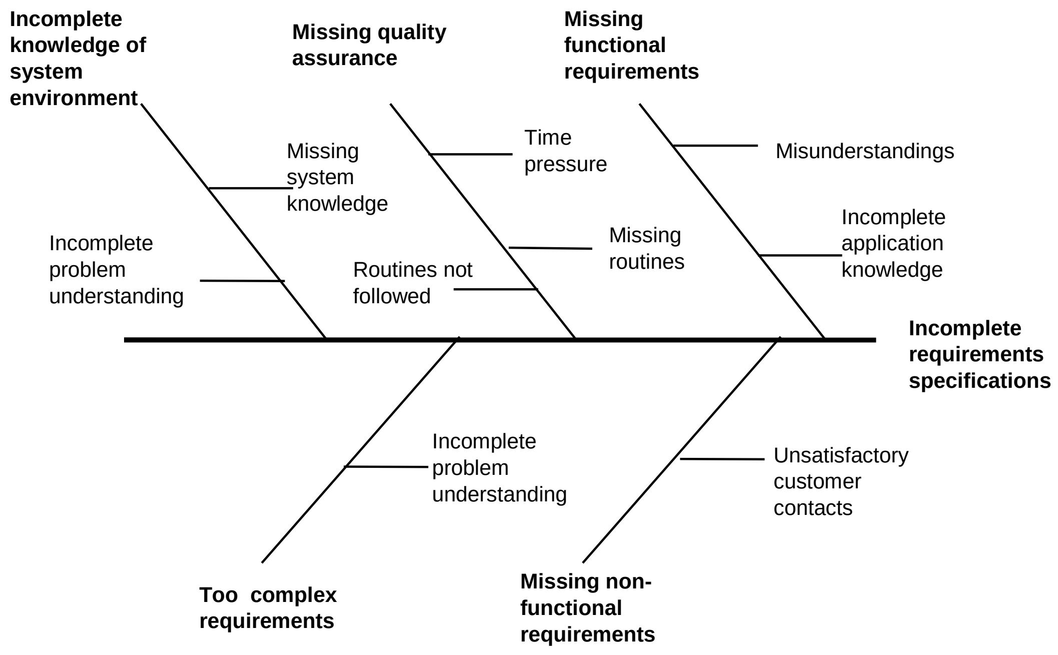

Incomplete knowledge of system environment
Missing quality assurance
Missing functional requirements
Missing system knowledge
Time pressure
Misunderstandings
Incomplete problem understanding
Missing routines
Incomplete application knowledge
Routines not followed
Incomplete requirements specifications
Incomplete problem understanding
Unsatisfactory customer contacts
Too complex requirements
Missing non-functional requirements

Figure 5 An Ishikawa diagram for “Incomplete requirements specification

Even if we use the Ishikawa diagram to identify possible improvements, the risks rising from changes to the process still have to be controlled. As demonstrated by others, there is a close connection between the GQM abstraction sheet and a risk management table. This connection stems from the observation that the environment factors in the GQM abstraction sheet are the important risk factors in the project – see Figure 4. The environment factors are important for two reasons:

1. They can have a strong influence on the results of the project if or when they change
2. They influence the values of the data collected in the project. If the environment factors change, a large part of our relations will change also.

The influences are documented in the “Impact on Baseline Hypotheses”-quadrant in the GQM abstraction sheet – see fig 4. Some of the environment factors are under control of the company, at least in the long run. This goes for such factors as the experience and knowledge of their developers, which can be influenced by courses and other forms of training. As a general risk control approach during improvement, we use Table 2 where the identified risks are at least the factors entered in the Variation factors part of the GQM abstraction sheet.

Identified risks	Estimates			Causes	Possible actions
	Prob.	Cons.	Risk		

Table 2 A general risk control approach

Conclusion

The situation with fast changes and few data is relevant for all SMEs and it will not go away. Thus, we need to establish an improvement framework for continues learning, not unreflective adaptation to best practice. The framework that we propose is a cycle of change – based on data – and stabilization – based on standardization and institutionalization. This is the same as the improvement framework originally proposed by the TQM fathers.

It is our experience that the continues learning needed for process improvement is best facilitated through a sequence of group processes. This sequence starts with the group process needed to define what to measure and how to measure it, goes on through collection and interpretation of data and ends up with institutionalization.

As a further result of few data and fast changes, we will seldom get enough data for solid statistical analyses. As a consequence of this, human knowledge and experience is of utmost importance when interpreting the collected data. Statistical analyses will, however, be important for deciding the significance level for our conclusions. Statistical analyses will also be important for areas where we will get large amounts of data. This is the case for process steps that are repeated often, such as code reviews, unit tests and maintenance activities.

An important alternative to experimental learning – at least in the strict sense of this word – is to go for direct improvement. This is done by combining Pareto analyses – what are our major problems – and one or more brainstorming sessions supported by an Ishikawa diagram – what can we do about it. This way of improving the process is faster, but carries a larger risk that the strict learning / understanding approach.

This view will partly collide with the idea of experience reuse. As stated several times in the past, one of the goals for process improvement is to collect data that can be stored in a data bank for later reuse. There are, however, some problems with the reuse of experience when we are strongly dependent on expert judgement. All the experts' knowledge stems from the process as it was before the improvement steps and is thus not reliable for the new, improved process. Reusable experience must thus focus on how to solve problems and on the parts of the process that were not changed.

However, the most important experience to reuse is that it is possible to improve the process through data analyses and the use of simple problem solving techniques. In all case, improvement should be attempted through several, small steps, not in one or a few giant leaps. The search for a best practice all too often results in a static view of the process, which is dangerous in an ever-changing market – at least as seem from the SMEs. In addition, it is important to keep in mind that static knowledge is not necessarily a good thing in a dynamic environment.

References

[1] Argyris, C., D.A. Schön Organizational Learning II, New York: Addison-Wesley, 1996

[2] Deming, W.E., Out of the crisis, Cambridge University Press, 1982

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

[4] Basili, V.R., Software Development: A Paradigm for the Future, in *Proceedings of the 13th Annual International Computer Software & Applications Conference (COMPSAC),* Keynote Address, Orlando, FL, September, 1989

[5] Basili,V. R. et al, The Software Engineering Laboratory – an operational Software Experience Factory, in *Proceeding from the 14th ICSE*, 1992

[6] Fenton, N.E., Pfleeger , S.L., Software metrics – A Rigorous & Practical Approach, International Thomson Computer Press, 1997

[7] Fiol, C.M. and M.A. Lyles Organizational Learning, Academy of Management Review, Vol. 10, No.4, pp. 803-813, 1985

[8] Huber, G.P.,Organizational learning: The contributing processes and the literatures, in *Organization Science*, Vol. 2, No. 1, February, pp. 88-115, 1991

[9] Kolb, D.A. Experiential Learning: Experience as the Source of Learning and Development. Englewood Cliffs, New Jersey: Prentice Hall, 1984

[10] Juul Wedde, K., Analysing metrics data, in *SPIQ technical memo*, 1997 (in norwegian)

[11] Lewin, K. Field Theory in Social Sciences, New York: Harper and Row, 1951

[12] Nonaka, I. and H. Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, New York: Oxford University Press, 1995

[13] Ould, M., SPC – mistake of the 90s?, in *Software Reliability and Metrics Club,* May 1997

[14] Perfect Consortium, Goal-Oriented Measurement Using GQM, in *D-BL-GQM-2-PERFECT9090, ESPRIT Project 9090 ”Perfect”,* 1997

[15] Perfect Consortium, Perfect Improvement Approach, in *D-BL-PIA-2-PERFECT9090, ESPRIT Project 9090 ”Perfect”,* 1997

[16] Polanyi, M.,The Tacit Dimension. New York: Doubleday, 1966

[17] Pulford, P., Kuntzmann-Combells, A., og Shirlaw, S., A quantitaive approach to software management - The ami handbook, Addison-Wesley Publishers, Wokingham, 1996

[18] Straker, d., A toolbook for Quality Improvement and Problem Solving, Prentice Hall, 1995

[19] Uchimaru, K., Okamoto,S., Kurahara, B.,TQM for technical groups – Total Quality Principles for Product Development, Productivity Press, Portland, Oregon, ISBN 1-56327-005-6

[20] Zultner, R. E. TQM for Technical Teams, in *Communication of the ACM*, pp. 79-91, Vol. 36 No. 10, October 1993

SPI in Embedded Software Applications

Bjarne Månsson
Software Group Manager

BARCO Communication Systems Denmark

BARCO Communication Systems AS

BARCO Communication Systems (BCS) has 3 divisions, each dealing with a specific range of products on broadcasting video and audio:

- BCS Belgium: Broadcast monitoring, cable TV systems, management systems.
- BCS North America: Digital transmission systems.
- BCS Denmark: Digital compression and transmission systems.

BCS Denmark

BCS Denmark has a long tradition for handling audio and video signals. In 1980, the company RE Technology based its business objective on audio test and measurement equipment, which in 1989 changed into telecommunication PCM transmission equipment (34 Mbit/s, 140 Mbit/s). In 1992, the know-how in audio signals and in telecommunication combined into audio and video broadcast communication before, in 1997, the company became a profound member of the BARCO Communication Systems.
The main products in BCS Denmark are codecs: digital video and audio compression equipment for high-quality transmission via common telecommunication networks.

The high-quality transmission usually relates to primary contribution of video and audio

- between the scene of recording and the studio,
- between studios, and
- between the studio and the transmitter.

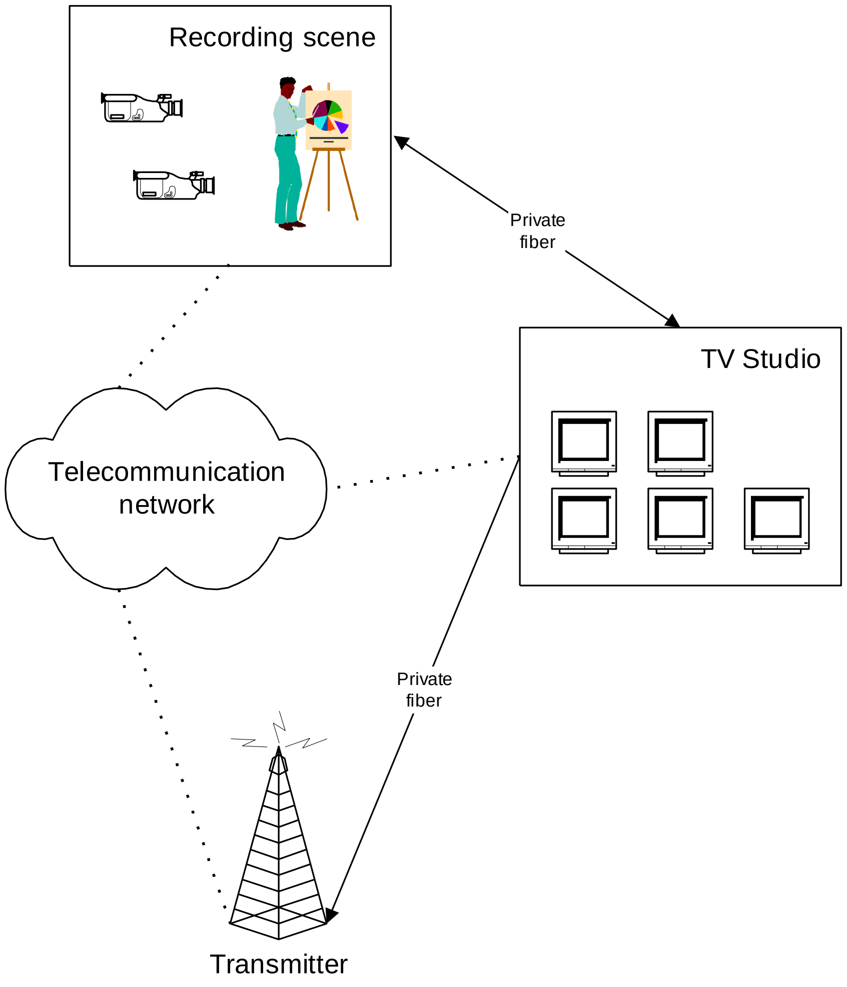


Fig. BjM.1 : BCS product application

Product techniques

A modern video codec features high-speed conversion of video signals at 300 Mbit/s into compressed video at telecommunication bandwidths of 45 Mbit/s, 34 Mbit/s, and down to 8 Mbit/s (MPEG-2).
A corresponding audio codec features conversion of audio signals at 384 kbit/s into compressed audio at telecommunication bandwidth of 64 kbit/s.
Because of the high requirements to conversion rates, the codec contains a lot of dedicated electronics in the form of ASICs and FPGAs.
Embedded software applies system control and monitoring but only signal processing at low bit rates (audio).

PC software is used to replace the equipment display and keyboard featuring equipment control and monitoring.

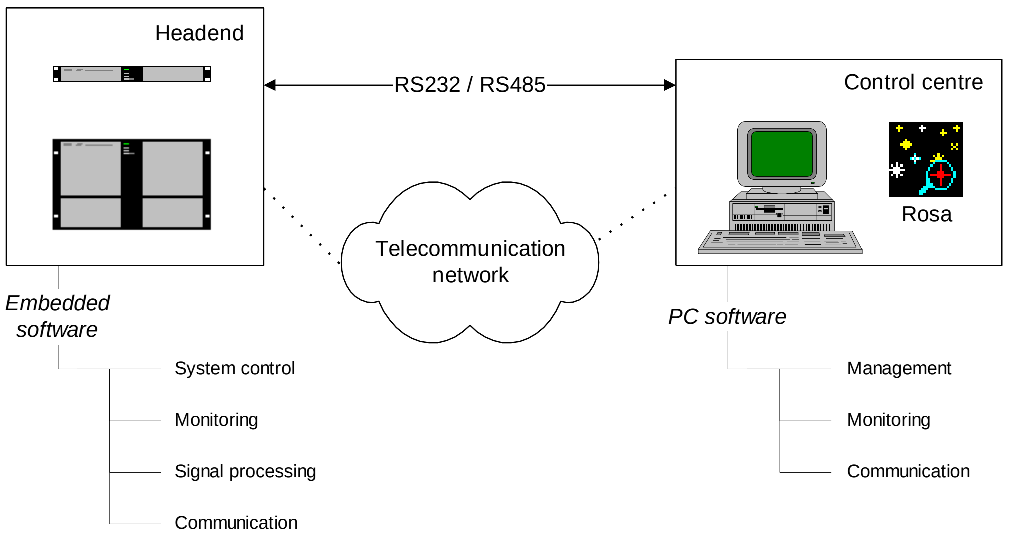


Fig. BjM.2 : BCS product software techniques

The initial problem

During 1994, the BCS released a major codec product, the RE 3400 ETSI video codec. The development of the codec had involved a project team greater than ever experienced during the past history of the company. In order to cope with a project of this size, a product life cycle model had been set up before the start of the project.

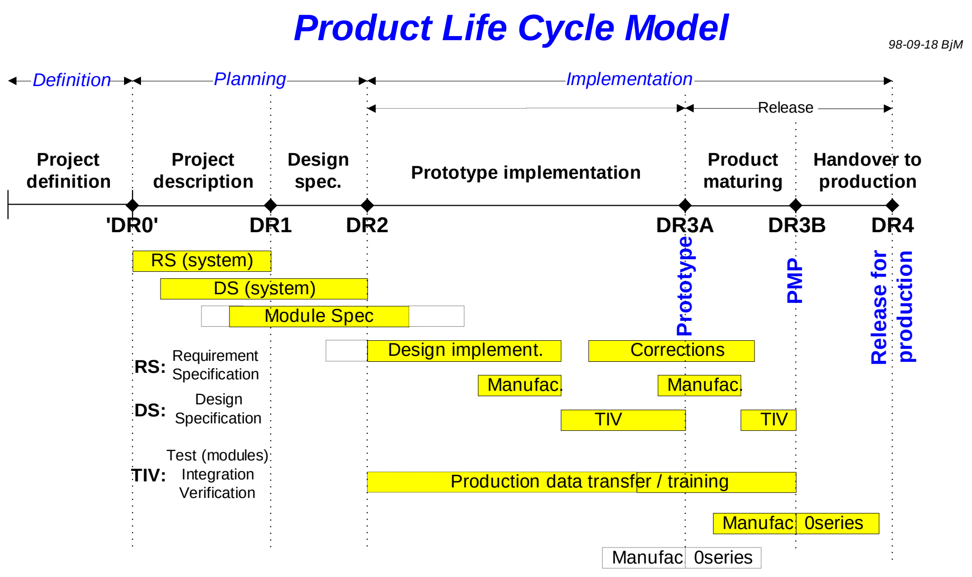


Fig. BjM.3 : BCS product life cycle model

The software crisis

Though the product development complied with the life cycle in all phases, the first releases experienced a number of drawbacks like:

- Releases were delayed because of incomplete software.
- Every release had a number of software errors (known as well as unknown).
- A new release had not always corrected old software errors.

The customer reactions were even worse:

- "The codec is not even starting when I apply power!"
- "Really, did you not see these fatal errors during your test?"
- "Is there NO work-around to this problem?"

The ISO 9001 issue

During 1994, the BCS thought it would like to be ISO 9001 certified. At the preliminary auditions by the certification institute, the software issue was brought up. The QA procedures were based on the product life cycle model and as the product was mostly hardware based, the software was only dealt with in 5 lines of text!

SPI initiated

The BCS management got very concerned at the alarming reports on the delayed codec releases and on the software problems obviously causing the delays and the bad quality. Furthermore, the road to the ISO 9001 certification was blocked by insufficient software procedures.

Another hint was given. The BCS had always promoted the use of the latest development techniques, "the-state-of-the-art". When somebody told the management that on a maturity scale from 1 to 5, BCS was on level 1 (<u>not</u> telling, however, that so was the situation for 90% of other companies too), an unsatisfied roar rolled through the company:

Do something about that software!

The SPI task force

Out of the roar came the establishment of a software group consisting of a newly employed group manager and the four present software engineers. The initial task was to evaluate answers to the outstanding questions:

- Releases were delayed because of incomplete software.
  Answers:
    - The software is specified too late in the product life cycle.
    - The software engineers are outnumbered compared to the hardware engineers (1:10).
    - The project manager is a hardware engineer.
- Every release had a number of software errors (known as well as unknown).
- A new release had not always corrected old software errors.

  Answers:
    - The software is not properly specified.
    - The software is not separately tested.
- "The codec is not even starting when I apply power!"
- "Really, did you not see these fatal errors during your test?"
- "Is there NO work-around to this problem?"

  Answers:
    - The software is not properly integrated with the hardware.
    - The codec product is not properly tested after the implementation of the software.
- The ISO 9001 procedures.

  Answers:
    - The software must be part of the (hardware) product life cycle model.

These answers were presented to the management who instructed the task force to set up an action plan of how to solve the problems. With the spirit of that time, the objective of the task force was expanded with:

Better software in a shorter time at a lower price!

Where to start

In my 20 years of software development, I have been looking for some way to get hold of this unpredictable, shapeless, intangible workmanship which somebody even has dared to call art. Though everybody knew of the problems, no firm philosophy, method, or tool had emerged though a few attempts had been performed, ref. [1]. This is very strange taking into consideration that hardware development is performed under well known methods as described in the product life cycle.

And then it starts as “best practice” out of the experience from many software developments’ “seek and try”. The inspiration came from the Capability Maturity Model CMM presented in Denmark during the spring of 1995, ref. [2].

Level	**Management**	**Organizational**	**Engineering**
1: Initial			
2: Re-peatable	Project planning Requirement Management Quality Assurance Configuration Management Project Tracking Subcontract Management		
3: Defined	Intergroup Coordination Integrated Software Management	Process Focus Process Definition Training Program	Software Product Engineering Peer Reviews
4: Mana-ged	Quantitative Process Management		Software Quality Management
5: Opti-mizing		Process Change Management Technology Change Management	Defect Prevention

Fig. BjM.4 : The CMM key process areas

Buzzwords arise all the time, and in 1995 everybody said “OOM”, “CASE tool”, and “Reuse”. The CMM indicated with which key process areas to start. It is e.g. pointed out that no method or tool can solve the lack of requirement specifications, and that in order to gain benefit from reuse, configuration management must be introduced.

Later on, we did not strictly follow the order of areas in the CMM, but we went more for every topic, which is also the basic idea in the Bootstrap model.

Initial SPI

Being a development company, the BCS is very project oriented. In order to be understood by the management, the task force (software group) handled its task as a project. The <u>software process action plan 1995/96</u> set the outlines of the project with the following headlines:

- Objective: Ensuring a fast, effective and controlled software development with a stabilized high level of quality.
- Project items: CMM key process areas of level 2.
- Time schedule, resource plan and a budget.
- Resources: Every software engineer (on part-time).

In order to make the ISO 9001 certification possible, the software was introduced into the QA procedures by software guidelines, which give recommendations to the essential documentation of:

- Software requirement specification.
- Software design specification.
- Software test and verification.
- Software configuration control.

For the project RE 3400, which initiated this SPI, we recommended two actions for every new release:

- Software requirement specifications on new functionality.
- Software test and test reports on the product functionality.

Short time results

It is important for every project to have some immediate results showing that the project is doing progress.
The first obvious result was the ISO 9001 certification in mid 1995 in which software was an integrated part.
Another result was recordings of an improved error rate measured on the codec RE 3400 releases.

	Software errors			
Ver-sion	found in β-test	known in release	found after release	removed
			24	
1.5		24	17	
1.6	42	37	9	4
1.61	20	37		
1.62	10	36	25	10
2.0	34	21	5	40
3.0	39	13	2	13
3.1	13	15	6	0
3.11	1	15	0	6
4.0	28	9	1	6
3.12	3	10	1	0
	190		90	80

Fig. BjM.5 : The RE 3400 error rates on releases

SPI continued

On obtaining the first results, we followed the same road when continuing the SPI:

- SPI action plan 1996/97.
- SPI action plan 1997/98.

We were then leaving the strict division into CMM key process areas and focused on the two key subjects of:

1. Software development methods and tools.
2. Software project management.

Development methods and tools

Despite good experience in writing software requirement and design specifications, we were still having difficulties in revealing all the relevant requirements from the hardware to the software. Though everybody still said "OOM", we were advised in real time applications to go for the Structured Analysis and Design. This was introduced in late 1995 together with the tool Select Yourdon, ref. [3] and [4].

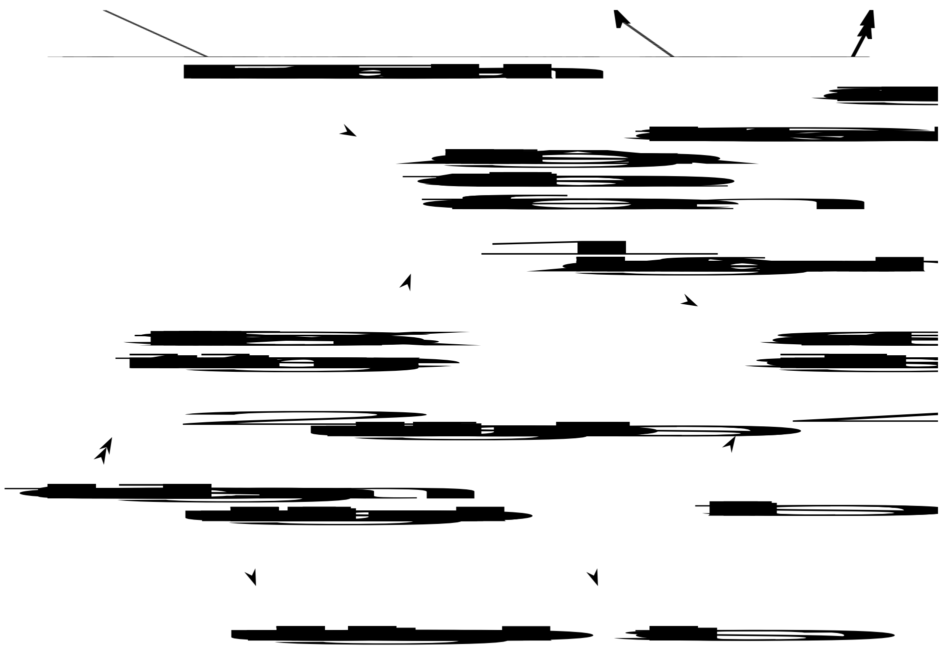


Fig. BjM.6 : Example on SA/SD-RT

The original object of the SA/SD-RT was for the hardware engineers and the software engineers to have a common base to discuss the full implementation. But it turned out to be a method for the software engineers to find all relevant questions in connection with the requirements.

Introducing the SA/SD-RT as early as in 1995 has enabled us to revise the method and the tool to our specific use. An example is that because we are not using entities (database data flows), we have removed this item from our recommended templates.

In the software implementation phase we had to set a number of rules:

- Only one compiler (Borland C)
- Only one linker/locater (Paradigm)
- Only one emulator (CheckMate)

These strange rules originate from the fact that previous software projects were isolated from each other. Having one software engineer on each of these projects, they had "succeeded" into choosing a different compiler to each project.

We could now see that the oncoming projects required a number of software engineers, which trigged off some more guidelines:

- Programming guidelines
- Version control (Intersolv PVCS)

```
/************************************************
*  Project      : Demoproject for PVCS
*  Used in      : PVCS demonstration
*  Description  : The module only contains a
*                 demo description.
*               :
$Workfile: demo.c $
$Log:  F:/sw_faggr/Projdemo/Source/vcs/demo.c_v  $
*
*    Rev 1.0   18 Feb 1997 13:34_08  BjM
* Description of the change in this revision
*
************************************************/
```

Fig. BjM.7 : Example of programming guidelines (program header)

The software test and verification is a subject too often put aside. In our case, we had already some very bad experiences of not properly testing the software together with the hardware. Choosing the V-model was very natural taking into consideration that the hardware already followed this model, ref. [5].

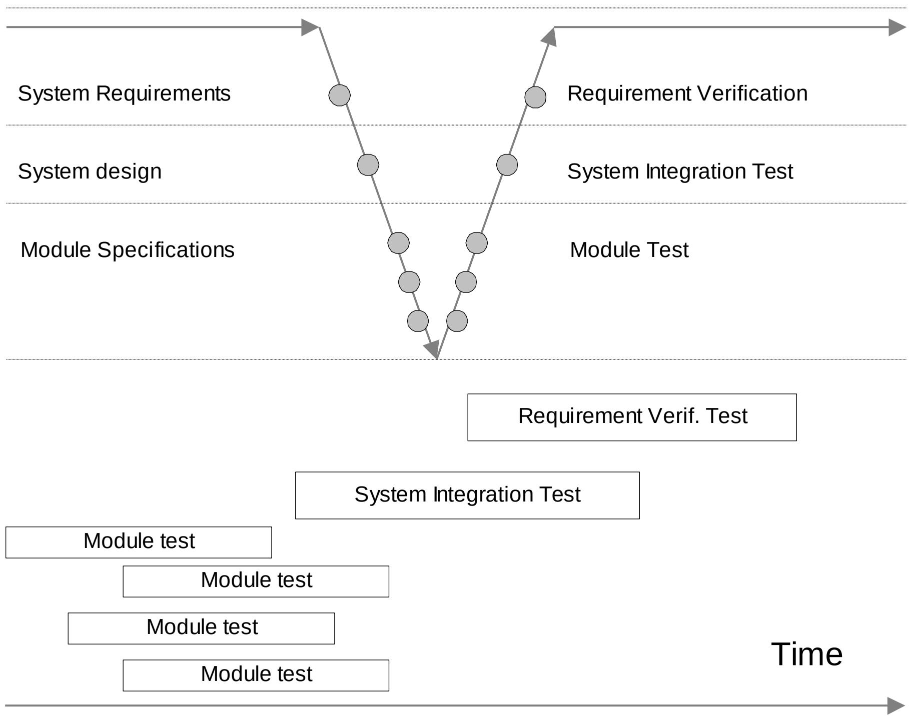


Fig. BjM.8 : The test and verification V-model

But finding the proper method and tool to each of the V-model stages turned out to be a difficult task. Software engineers were used to test the software in the "monkey" way - testing what they thought should be tested which is not much more than usual debug testing.

After some research, we recommended the test following methods and tools:

- Code review (checklist)
- Static analysis (Borland C compiler warnings)
- Dynamic analysis (IPL Cantata)
- Test cases (template)

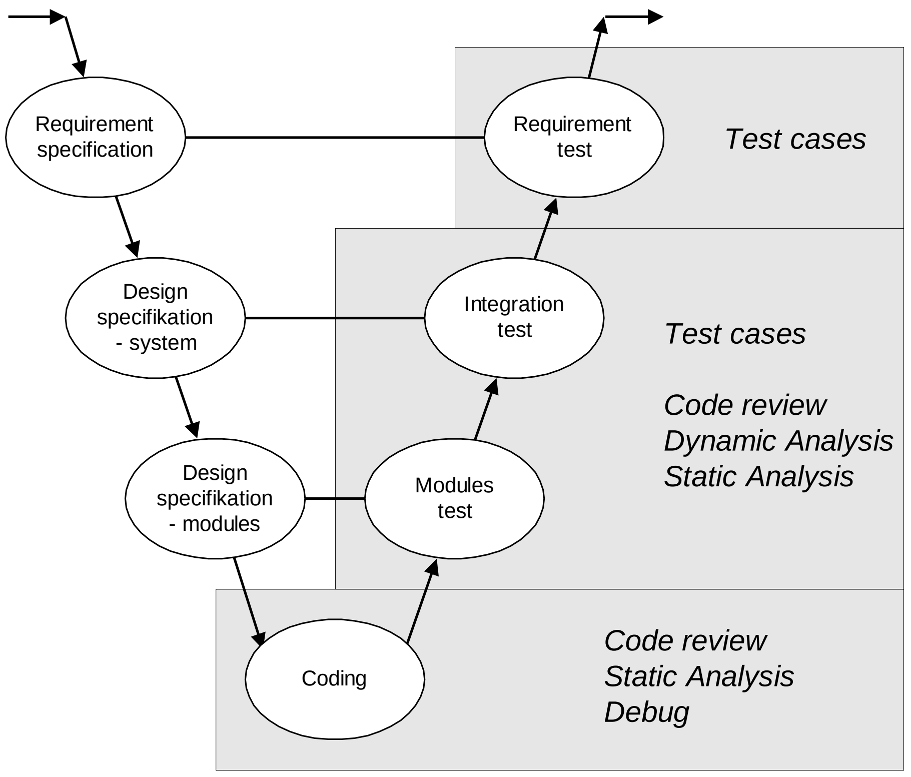


Fig. BjM.9 : Software test tools

One hurdle to overcome was the strong belief of every software engineer that a test tool can do the methodical work too! "Is the tool not able to generate the test case for me?" No, there is still a lot of test work to do for the software engineer.

Project management

Software project management has the same contents as hardware project management. In the beginning, the project managers (being hardware based) did not believe this. But when we produced a project management guideline for using software in hardware, they started to be convinced, ref. [6].

The management guideline contains the following subjects of:

- Project creation.
- Planning.
- Reviews.
- Follow up.
- Metrics.

When we were filling in descriptions of how to perform these subjects, it became more or less a repetition of what we already had described for the “product” project management (the ISO 9001 QA procedures).

As to the <u>software project creation</u> we confirmed the following items:

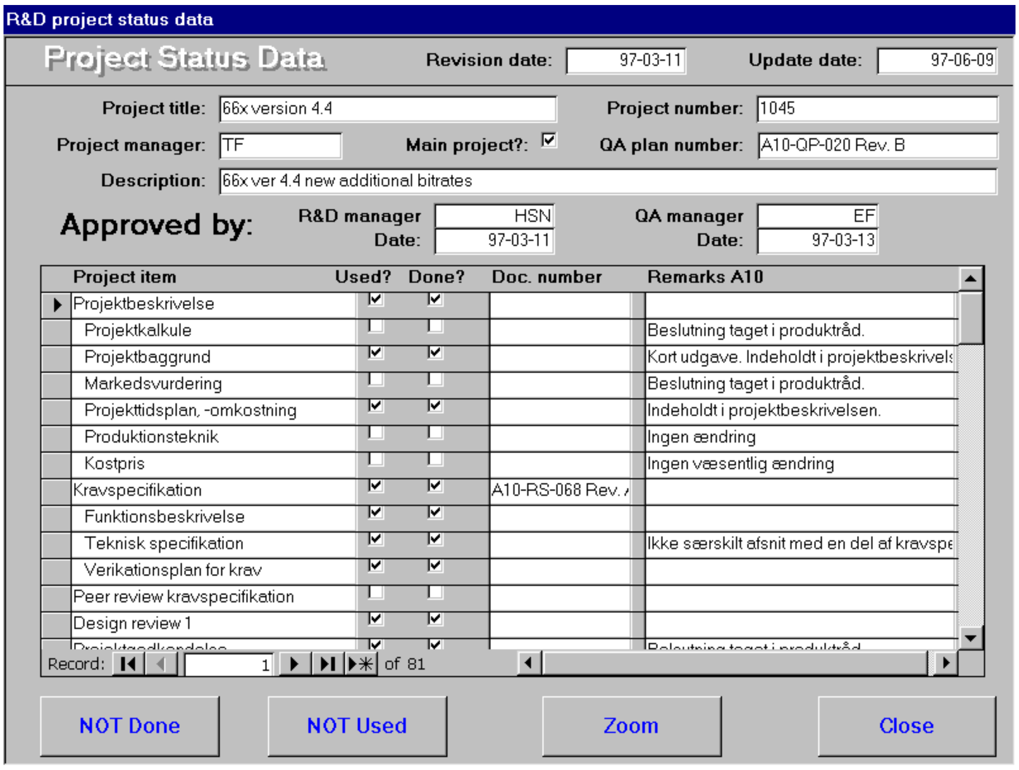
- ISO 9001 QA procedures: how to develop while ensuring proper quality
- Product life cycle model: the phases of the project
- Risk analysis: pinpointing the major risks in the project

No.		Poss. 0,2..0,8	Effect 1..10	Weight P*E	Preventive actions	Prepared actions	Signals
1,0	**The product**						
1,1	Is the product technically wrong? - Are the technical requirements difficult? Is the product the-state-of-the-art?	0,4	10	4	Check with Barco		
1,2	Is the product wrong for the market? - Is the market moving? Do we know the market?	0,8	5	4	Check with marketing communications		Customers decline use of management network
1,3	Is the quality too bad? - Do we usually see many errors after release?	0,2	8	1,6			
2,0	**The frames of the project**						
2,1	Are the goals and subgoals unclear? - Are the goals too ambitious?	0,2	10	2			
2,2	Is the project description still unsettled by DR1? - Is the requirement specification well prepared? Is the project plan by DR1 realistic?	0,5	5	2,5			
2,3	Inadequate resources? - Do you expect additional resources during the project? Do you expect overtime work?	0,8	5	4	Commitment from Product Council. Agreement with Barco	Asking Barco for resources. External resources	Increasing delays on START of items

Fig. BjM.10 : Example of risk analysis

As to the software project planning we confirmed the following items:

- Estimation: Still based on the experience of the individual engineer.
- Project plan: MS Project tool.
- QA checklist: The 81 QA procedure checkpoints required during the project.



R&D project status data

Project Status Data

Revision date: 97-03-11 Update date: 97-06-09

Project title: 66x version 4.4 Project number: 1045

Project manager: TF Main project?: ✓ QA plan number: A10-QP-020 Rev. B

Description: 66x ver 4.4 new additional bitrates

Approved by: R&D manager HSN Date: 97-03-11 QA manager EF Date: 97-03-13

Project item	Used?	Done?	Doc. number	Remarks A10
Projektbeskrivelse	✓	✓		
Projektkalkule				Beslutning taget i produktråd.
Projektbaggrund	✓	✓		Kort udgave. Indeholdt i projektbeskrivels
Markedsvurdering				Beslutning taget i produktråd.
Projekttidsplan, -omkostning	✓	✓		Indeholdt i projektbeskrivelsen.
Produktionsteknik				Ingen ændring
Kostpris				Ingen væsentlig ændring
Kravspecifikation	✓	✓	A10-RS-068 Rev. /	
Funktionsbeskrivelse	✓	✓		
Teknisk specifikation	✓	✓		Ikke særskilt afsnit med en del af kravspe
Verikationsplan for krav	✓	✓		
Peer review kravspecifikation				
Design review 1	✓	✓		

Record: 1 of 81

Fig. BjM.11 : QA checklist

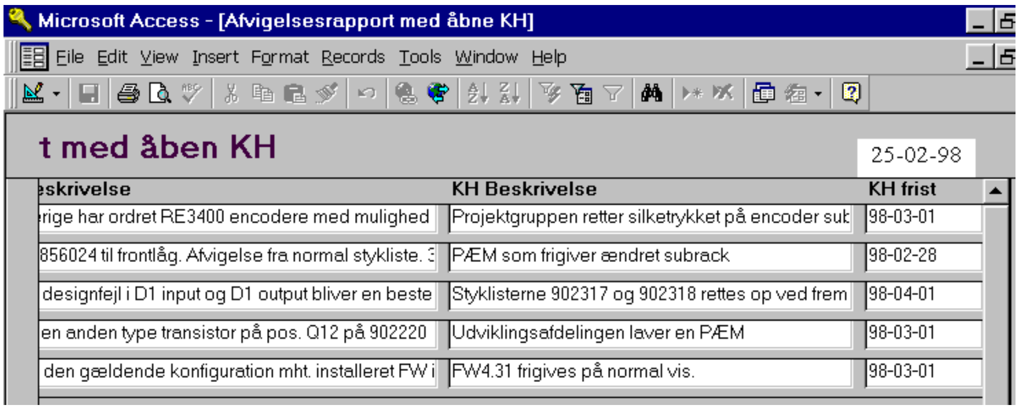
As to the software project reviews we confirmed the following items:

- Requirement specification: Check list on major items.
- Code review: Check list of reasonable items.

```
* Structured code?
* Requirements mapped into the code?
* Is the SD-RT diagram implemented?
* Is the interface to the module OK?
* Is the code easy to maintain?
```

Fig. BjM.12 : Code review checklist

As to the software project follow up we confirmed the following items:
- Project meetings: Minutes of meeting with action list.
- Monthly report: Project progress and problems.
- Release plan: Monthly revision of the product release dates.
- QA deviation report: All deviations from the QA procedures are reported.



t med åben KH

25-02-98

›skrivelse	KH Beskrivelse	KH frist
rige har ordret RE3400 encodere med mulighed	Projektgruppen retter silketrykket på encoder sub	98-03-01
856024 til frontlåg. Afvigelse fra normal stykliste. 3	PÆM som frigiver ændret subrack	98-02-28
designfejl i D1 input og D1 output bliver en beste	Styklisterne 902317 og 902318 rettes op ved frem	98-04-01
en anden type transistor på pos. Q12 på 902220	Udviklingsafdelingen laver en PÆM	98-03-01
den gældende konfiguration mht. installeret FW i	FW4.31 frigives på normal vis.	98-03-01

Fig. BjM.13 : Example on QA deviation reports

As the software project metrics we confirmed the following items:
- Time statistics: Development time used on each module.
- Evaluation: Project history, major differences to the original project plan.
- Quality reports: Monthly reports on released products.
- Product problem reports: Customer reported problems.
- Release plan tracking: How close did we get to the planned milestones?

Milestone	**Hit rate (on time or better)**	**Hit rate (delay ≤ 20%)**
Prototype milestone	45 %	60%
Product matured milestone	35 %	55 %
Release milestone	30 %	45 %
Milestone differences		
From prototype to product matured	65 %	70 %
From product matured to release	75 %	80 %

Fig. BjM.14 : Release plan tracking

In BCS, the software metrics has been an overlooked subject, so even if we did introduce these items to the projects, we could not retrieve information from the "experience" database. And because we did not get any immediate results out of the metrics, the trend was that we did not even do any metrics on the new projects.

In summary we did not introduce any new software project management items, but we merely confirmed the existing hardware related items. In a few cases, e.g. risk analysis, we expanded the items of the subject.

SPI assessment

One of the things that started the SPI in BCS was the allegation of our maturity level being level 1! Naturally, a metric of the SPI project is the measure of maturity level.

As to the action plans, we set the goal of reaching a specific level:

- Action plan 1995/96: Level 2
- Action plan 1996/97: Level 2.5
- Action plan 1997/98: Level 3

But how do we measure our maturity level? In the CMM, you can only be on one level of 1, 2, 3, 4, or 5, and only if you are complying with all key process areas of that level. Doing any improvements of a higher level does not count on a lower level.

This is handled in the Bootstrap model, which compiles all process improvements into a single number with divisions of a quarter. In this way, even minor improvements can be measured, and you benefit from measuring even small improvements, which is good for the motivation, ref. [7].

The Bootstrap assessment can be performed by either self-assessment or by certified assessment.

The self-assessment involves a questionnaire which one or more people from the organisation may answer.

The certified assessment involves 3 days' interview of management and of a number of project groups made by external auditors.

At the end of 1996, BCS decided to do both a self-assessment and a certified assessment. The self-assessment was based on three different questionnaires and it gave the following results:

Bootstrap assessment method	Level
BOOTCHECK (46 questions)	2.5 - 3.5
ESSI committee (43 questions)	2.5 - 3.0
SYNQUEST (370 questions)	2.0 - 2.5
Certified ("3 days of questions")	2.3 - 2.5

Fig. BjM.15 : Assessment results

The questionnaire containing most questions (SYNQUEST) gets the nearest to the certified assessment (luckily enough!), ref. [8]. But anyhow, any self-assessment gives a clue of the present maturity level:

Better do some self-assessment than none at all!

SPI summary

You get most benefit from SPI if you apply the improvements on concurrent projects.

We applied the SA/SD-RT and implementation methods on 3 major projects with good results. But the introduction of the test and verification V-model was delayed compared with the progress of the 3 projects, resulting in a very bad software quality on one of the major projects.

Also, software project management was not introduced early enough on the 3 projects. The software time planning (time estimates), software project plan, and the risk analysis (all due in the early phase of the project) were hardly used in the product project.

And now we have to wait for the next major project before we can benefit from these methods!

An advantage of introducing SPI as part of the QA procedures was that we were not delayed by any “pilot project evaluation”. Though we did some trial investigations before introducing a major new method or tool, we gained a lot of knowledge and motivation from everyone being educated to the same level at the same time.

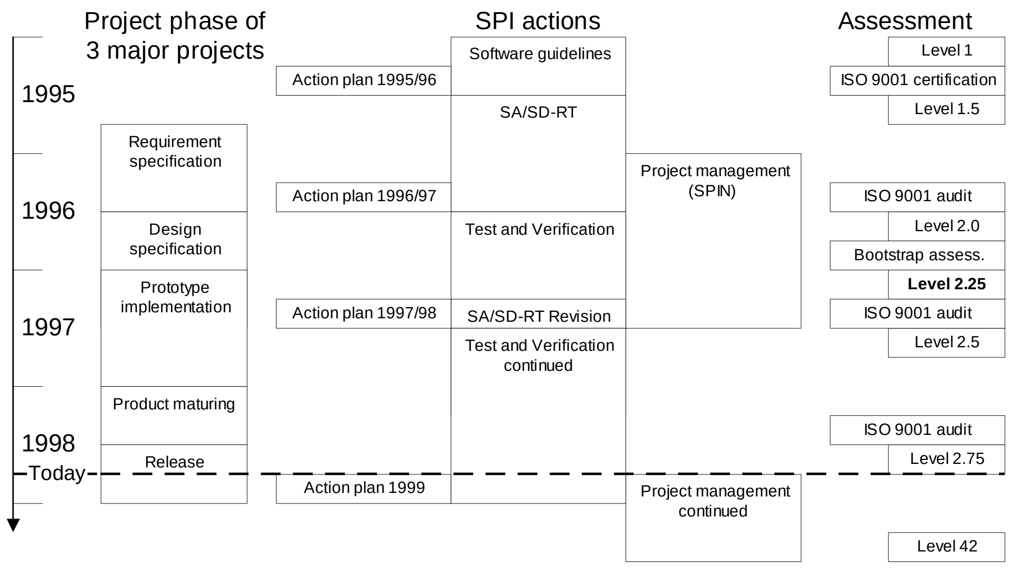


Fig. BjM.16 : SPI Summary with respect to Concurrent Projects

SPI lessons learned

Introducing SPI in a development-based company requires a strong health and lots of good spirits - like in any other project matter!

But keeping some rule-of-thumb in mind can help you through the strongest of bad luck.

Below I sum up some of the main good and bad experiences from the 3 years of SPI introduction.

The SPI iceberg

Beware of the SPI iceberg. It is easy to "buy" a method or a tool but it is much harder to get it working inside the company.

- Implementation must be thoroughly planned.
- Management must commit itself to the SPI.
- You yourself must urge a publicity drive to "sell" the SPI to the projects.

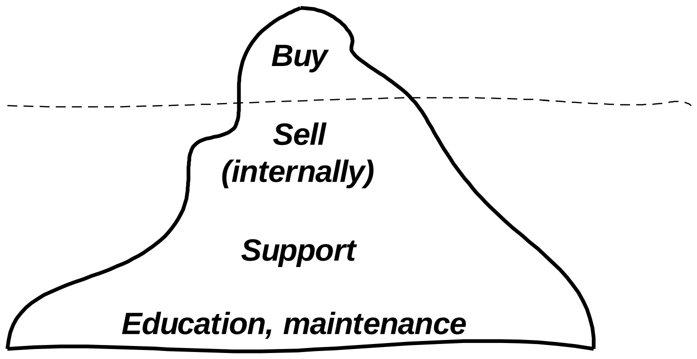


Fig. BjM.17 : The SPI iceberg

The SPI "silver bullet" life cycle

Beware of the SPI "silver bullet" life cycle, ref. [9]. It is easy to set up unrealistic expectations to the outcome of the SPI, but that only works until the SPI is being used in real life. The hard work is to get SPI down to earth and get it working in actual practice.

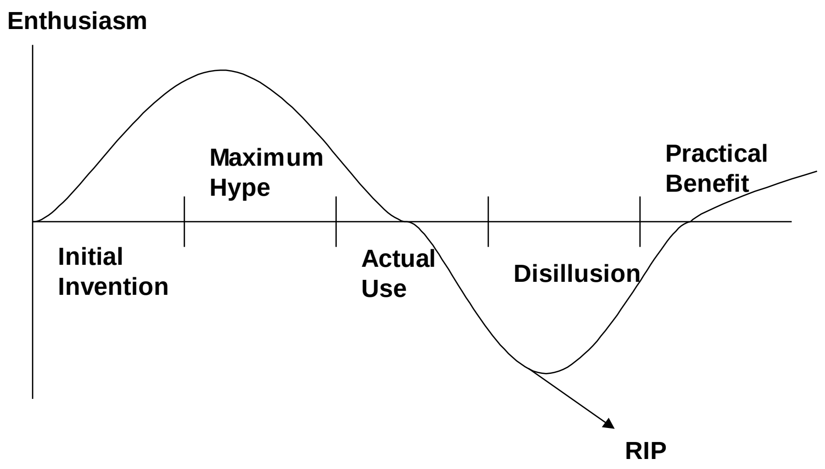


Fig. BjM.18 : The SPI "silver bullet" life cycle

The SPI good experience

- Software is now an equal member of the good party
- We have grown from 5 to 20 software engineers and we still have a common development basis
- Synergy (and reuse) between 3 major projects
- SPI on all projects motivate people
- Better software delivery time
- Presumably better software quality
- We know how good we are!
- And we know our shortcomings!

The SPI bad experience

- SPI mental process takes time (2 years per level)
- SPI motivation curve has still not turned into positive
- Project managers are not yet properly educated
- Software module test is not properly introduced
- Software metrics are absent - no experience database has been established
- SPI maintenance takes its toll
- It has not become cheaper to develop software
- At the moment we are not becoming better

SPI in future

Still, many SPI subjects and items have to be fulfilled - and changed! This will be when new projects require revised methods and tools!

For the next action plan, the main topics will be:
- Keep the same software group setup.
- Fill in the gaps found, Overcome pitfalls found.
- Turn the SPI motivation curve into positive.
- Define the proper end maturity level (3?).
- And maintain it.
- Synchronize with the hardware process.

BCS software began as a supplier to the hardware but this is not the case anymore. We must look upon each other as equals in the cause of product development.

Abbreviations

CASE:	Computer Aided Software Engineering
CMM:	Capability Maturity Model
OOM:	Object Oriented Model
SA/SD-RT:	Structured Analysis and Design in Real-Time
SPI:	Software Process Improvement

References

[21] [1] S. Biering-Sørensen, F. Overgaard Hansen, S. Klim, P. Thalund Madsen: Håndbog i Struktureret Programudvikling, Teknisk Forlag, 1988.

[22] [2] Capers Jones: The Path to Software Excellence: Becoming “Best in Class”, SPR, Inc., March 10th, 1995.

[23] [3] P. Ward, S. Mellor: Structured Development for Real-Time Systems, Prentice-Hall, 1984.

[24] [4] F. Overgaard Hansen, F. Hansen: SA/SD-RT Kompendie 940131, DTI, Århus, 1994.

[25] [5] Boris Beizer: Software Testing Techniques, Van Nostrand Reinhold, New York, 1990.

[26] [6] W.S.Humphrey: Managing the Software Process, Addison-Wesley, Reading, MA, 1989.

[27] [7] P. Kuvaja, et al.: Software Process Assessment and Improvement – The Bootstrap Approach, Blackwell Business, Oxford, 1994.

[28] [8] SynQuest, version 1.5: Selfassessment for Softworkers, SynSpace GmbH, 1996.

[29] [9] F.P. Brooks: No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, Vol. 20, No. 4, April 1987.

Appendix: The author and the company

Bjarne Månsson

With more than 20 years of software background, Bjarne Månsson has experienced the need of and the requirements to the SPI movement. He graduated in 1974 with a M.Sc. degree from the Technical University of Denmark (DTU), which in 1979 was extended with a M.Phil. Degree from the University of Leeds, UK. Starting in 1976 in the telecommunication world, developing test equipment for telephone exchanges, he joined the first attempts to embed software in purely hardware-based products. His knowledge in this field was widened in much greater scale during the 1980s where he worked as project manager of data acquisition equipment including data collection electronics and data processing mainframe software. After a short visit to the CNC machine industry in the early 1990s, also developing data acquisition, Bjarne Månsson returned to the telecommunication business, being responsible for the introduction of SPI in embedded software in high quality broadcasting electronics.

BARCO Communication Systems Denmark

The BARCO Group's main business area is projection systems, which covers one third of the group sales. The closely related display systems and graphics systems cover another third of the group sales.
A BARCO group member is the BARCO Communication Systems, which is a world leader in high quality solutions for the broadcast, cable TV and telecommunication markets with

- Broadcast display systems
- Broadband communication systems
- Broadcast and telecommunication networking systems

The BARCO Communication Systems has three divisions (Belgium, North America and Denmark) doing development, production, marketing and sales of

- Broadcast monitoring
- Digital compression systems
- Digital transmission systems
- Cable TV headend systems
- Operations support system (ROSA)

The BARCO Communication Systems is present world-wide with offices in Germany, France, UK, the Netherlands, Israel, USA, Mexico, Brazil, Hong Kong, China, Malaysia, Japan and Australia.

An Experience of SEPG Organization

Alessia Billi
Sodalia S.p.A., Trento- Italy
alessia@sodalia.it

Abstract

A key element of the Sodalia Software Process Improvement is the establishment of a Software Engineering Process Group (SEPG), that consists of members of the Sodalia Methodologies Area, completely devoted to activities related to processes and methodologies definition, and of representatives of the software development projects.
The group has been established with the objective of co-ordinating the improvement activities of the company software processes.
The approach to fulfill that objective is based on three key steps: identification of improvement actions, implementation of these actions by work groups, extension to the whole organization.
The achieved results show the benefits of this approach: the company obtained the certification ISO 9001 and has been assessed at level 3 of CMM in an outstanding time frame of about four years.

Introduction

The institutionalization of a Software Engineering Process Group is a key element in the improvement of the organization's processes capability, and in the increase of the overall company maturity.
The Capability Maturity Model at level 3 requests the existence of a similar group, in order to develop, understand, maintain and improve processes related to the software projects and to the organization.
The model indicates different suitable solutions for the organization of the SEPG, involving full-time or part-time resources assignment.

Company Context

Sodalia is a medium size company (200 technical staff) located in Trento (north of Italy), developing advanced software for the management of telecommunication services and networks.
It was established in 1992, as a joint venture between TELECOM Italia and Bell Atlantic Corporation, and started operations in 1993.
From its foundation, among the main company's mission statement, the objective of improving the quality level of products occupied a relevant position. In order to reach this objective, Sodalia decided to steer its effort both to improve the maturity level, according to the SEI Capability Maturity Model, and to obtain the ISO 9001 certification.
The company was assessed at Level 2 of the Capability Maturity Model in December '95, in May '96 received the ISO 9001 certification, and finally, in September '97, has been assessed at Level 3 of the SEI-CMM.

Improvement Steps

The model adopted by the company for the improvement of the processes is the Capability Maturity Model of the Software Engineering Institute.
As an important objective of the Company has been, since its foundation, the achievement of a satisfactory level of processes and products quality, a significant effort was spent in the definition of common process already while the company was moving its first steps in software development.
The first step of the company in this direction was the definition of the software development process model, called SIMEP (Software Integrated Management and Engineering Process), founded on the following basic principles: iterative incremental/evolutionary approach, software reuse, object-oriented approach, full integration of project management and software engineering activities.
After its first modeling, the process has been consolidated and the related documentation has been enriched; the improvement steps followed each another in ambit of selected improvement areas based on the Key Process Areas of the CMM.
For each improvement area, a set of guidelines and a templates have been defined, in order to make homogeneous the way of performing the various activities among different projects during the software life-cycle.
The topic of requirement management has been faced in several subsequent improvement steps in order to obtain a homogeneous classification and description of the product requirements.
Starting from the original textual way of requirements description, the improvement steps have been addressed to structure the requirements, by identifying each single requirement with the needed granularity, to classify the requirements in more categories and sub-categories and to define a status vector for each requirement in order to manage its status and its attributes.
The result is the currently used layout and the defined process for requirements definition, that guarantee that the requirements are managed and controlled and that the traceability is kept.
Also as regarding the project management, two improvement degrees have been obtained in two different, subsequent steps: at first the effort has been spent into the definition of common process and instruments for planning and controlling the projects; a subsequent improvement followed, aimed to support the project managers

in sharing the planning and monitoring company experience and in managing the project risks. In this phase, a specific improvement step has been devoted to the identification of all possible risk factors for the company projects, to be kept into account by the project managers during the risk planning.
Configuration management activities have been defined and supported for making the software products and intermediate artifacts managed and controlled.
The verification and validation of the products has been guaranteed by making reviews of process artifacts and software test performed by adopting standard process and method, and by defining the classification and weight of the defects.
Guidelines for process tailoring have been produced for supporting the process scalability: the process is currently adaptable to various project and product types, sizes and risk classes.
Data measurement and metrics constituted an important topic to be investigated; a set of guidelines has been defined and then made more usable by the creation of a *software process database* for an homogeneous collection of data and measures; the company projects data have made available to be used for estimation activities.
These important improvement activities strongly connected to the Key Process Areas of the CMM have been completed by a set of other improvement steps, related for example to the selection of particular tools or methods to support and perform the development activities.
After the assessment to the Level 3 of the CMM, most effort has been spent in the improvement of the standard set of company tools, and in introducing smooth changes to the company processes towards the Level 4 of CMM.

Organizational Problems Faced

The principal organizational structure that supported all the described activities has been, from beginning of the improvement activities, the Methodologies Area, composed of technicians fully devoted to the study of all the topics connected to the software engineering.
This group analyses the problems related to all the improvement items, investigates the best results available in literature and tries to adapt them to the company context, keeping into account the problems to be faced day by day by the people involved in the projects activities, due to the organization, the products domain and the type of the clients.
The existence of this Area played a fundamental role in the achievement of the main company–level results, as assessment of levels 2 and 3 of CMM and the ISO certification.
Nevertheless, the overall organizational approach to the software process improvement has been tuned and changed during the first 5 years of the company life.
The Methodologies Area staff in fact, having the objective of investigating in deep the various topics, gained a good level of knowledge about the theoretic aspects of the faced arguments; however the problems connected to the practical application of the methodologies and processes in a particular context and products domain are sometimes not immediately visible, and the contribution of the experience coming from the members of software development projects is invaluable.
Furthermore Sodalia experience showed that members of the projects, if not involved

in various decisions at company level, could suffer the technical decisions like impositions; as a consequence the methodologies risk to be "placed on the shelf".
But, at the same time, project staff feel the involvement in methodologies and processes definition as a lower level priority activity; in addition, the Methodologies Area has the commitment to guide the company improvement according to CMM as a reference model.
For these reasons an organizational approach calibrating the contribution of theoretical elements and practical experience has been considered necessary for guarantee the success of the company improvement steps.

The Organization of the Software Engineering Process Group

To fulfill this need the Software Engineering Process Group (SEPG in the following) has been established in July 1996, that guided the company improvement in the transition from level 2 to level 3 of CMM.

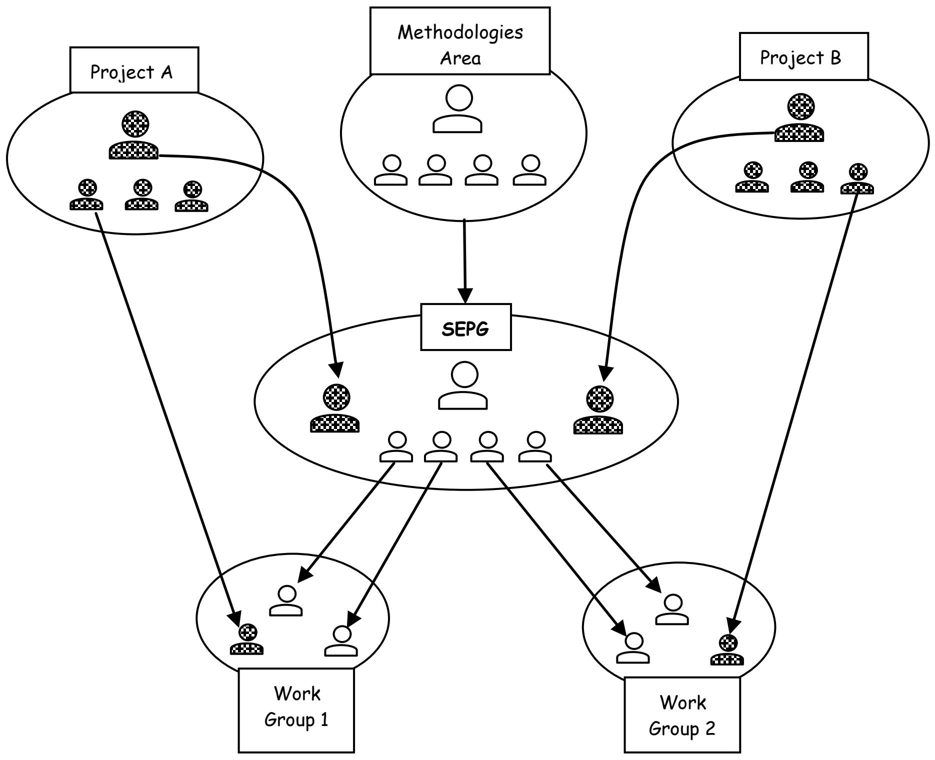


Fig. ABI.1 : SEPG and Work Groups Composition

The Group is composed as permanent members by the Methodologies Area

representatives and by the project managers of all the company projects. In addition, according to the particular needs, other company staffs are involved in SEPG activities for short periods (see fig. ABI.1).
The selected approach is based on the selection of improvement areas, with clear objectives, and on the implementation of related actions by the establishment of work groups (see fig. ABI.2).
This implementation is followed by the extension of the achieved results in the overall organization.
This approach includes points of centralization of both information and decisions, and points of delegation of the activities to specialized groups, optimizing in such a way the effort expended by all the participants according to their roles and capabilities.

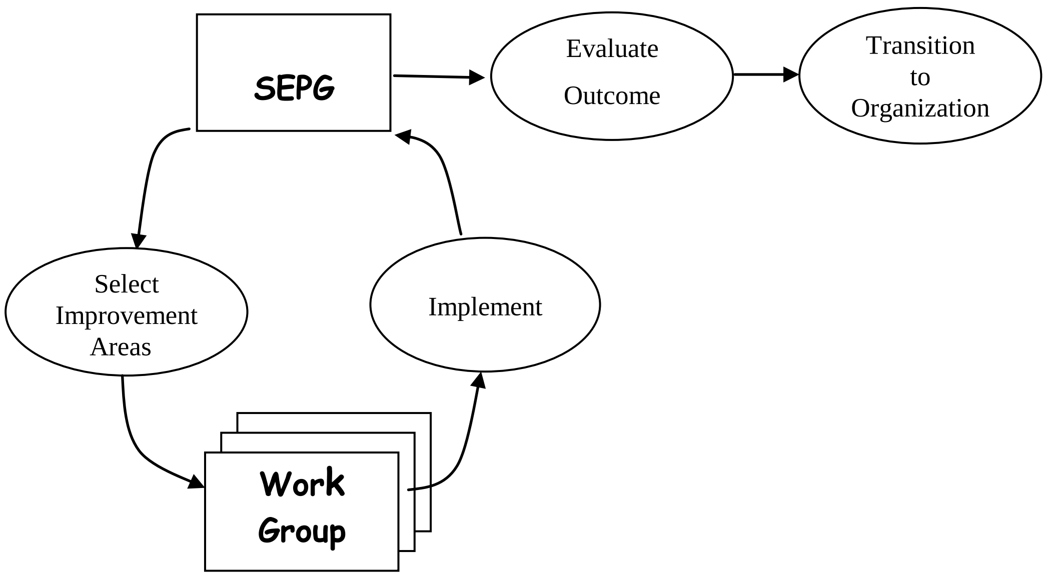


Fig. ABI.2 : SEPG and Work Group activities

The improvement areas are identified by the SEPG, on the basis of proposals suggested by either Methodologies Area staff or projects members; generally the proposals arise from assessments, from analysis of industry trends and emerging technologies or from project needs.
The work groups having the objective of implementing these areas are composed trying to involve people capable of give the maximum contribution: Methodologies Area staff with a strong theoretic knowledge and practitioners with technical responsibilities for certain domains within a development project and with a significant experience.
Methodologies Area staff are in charge for organizing and coordinating the work of these groups, and collecting, synthesizing and reporting the results.
As already noted, the involvement of the Methodologies Area is full-time, whereas

projects provide in turn temporary resources, so that the effort spent affects only in minimal part projects schedules.
The responsibilities of the work group consist of the following:

- focus on an aspect of the software process;
- proposal of improvement actions, defining the objectives and the expected benefits;
- implementation of the improvement actions, as assigned by the SEPG;
- support to projects.

The work groups' intermediate results are frequently presented to the overall SEPG, in order to maintain informed all projects representatives and involve them in the most important decisional issues, minimizing the involvement time.
At the moment of the improvement area implementation conclusion, the sharing of the information in the whole company and, most important, the agreement, are so assured, and the transition phase is immediate.
The sharing of the information on the SEPG activities is also facilitated by the support of the available tools as the Software Process Database and the use of the internal web as a communication mean.

An example of work group

Based on a proposal of the Methodologies Area, a specific session of SEPG meeting discussed the opportunity to improve the Sodalia development infrastructure, adopting a UML-compliant process supported by a single CASE tool.
The first emerged necessity was the evaluation of the Return of Investments: a first work group has been formed with the objective limited to this duty.
Four Methodologies Area members, two project architects (from different projects) and one project manager formed this work group.
The work group presented the results to the SEPG two months later.
After the evaluation of the results the SEPG committed two other work groups: one with the objective of analyzing and proposing the changes of the software process and documentation standard (suggesting a practical way to achieve common semantic and UML notation for specification and design phase), a second work group with objective of performing the screening of the candidate case tools, defining the way for performing its experimentation and reporting the results.
The first work group started working at the end of June 1998; three Methodologies Area members and the architects of all Sodalia projects composed it.
The group met about every ten days, and a Methodology Area member was responsible of reporting the collected and discussed information and formalizing it in a document, whose key elements were presented to the SEPG about every two months.
The activities of second group started at the end of July 1998.
Two Methodologies Area members, one architect, one project manager and three developers, working in different projects, formed the group.
The Methodologies Area members, being members also of the first work group, guarantee the compliance of the results of the two groups.
The agreement achieved and the presented results showed the success of the selected

organization approach.

Benefits of this approach

The institutionalization of this organizational approach, involving in different ways different company roles, compared with the organization adopted during the previous phases of improvement, shows evident benefits that could be summarized as follows:

- project managers and development staff are in turn fully involved in the selection and definition of processes, methodologies and tools, assuring the contribution of the practical point of view in this kind of activities;
- the systematic approach to the changes to be introduced in the company facilitates the activities and allows to reuse the matured experience;
- the natural resistance to the changes by the projects members, used to perform the activities in a traditional way, is minimized, due to the extended involvement in the decisional process;
- the information connected to each improvement phase is diffused in a short time;
- a communication point is established also for the sharing of best practices matured within a single project, with evident benefit for the overall company.

Conclusions

To support the analysis and the solution of problems related to the management of the software development processes and methodologies improvements, the help coming from people directly involved in the project environment is invaluable, but a centralized organization can play a very important role for assuring the sharing of experience. For this reason a mechanism for diffusing decisions and lessons learned is very important.
As a result, this approach assures the best exchange of knowledge, maintaining the correct balance between the theory and its pragmatic application.

References

[30] [1] Paulk M.C., Weber C.V., Garcia S.M., Chrissis M., Bush M., *Key Practices of the Capability Maturity Model*, Version 1.1, Tech. Rep. CMU/SEI-93-TR-25, Software Engineering Institute, Carnegie Mellon University.

Session 4 – Metrics driven SPI Part II

Introduction of Metrics in Civil Engineering Software Development

ESSI PIE No. 27272 - ICENSOM

Dr. A. Tsipianitis
TEGEA SA, Athens, Greece

SIMMER: Software and Systems Integration Modelling Metrics and Risks (Getting to Level 4)

Brian Chatters
ICL, Manchester, UK
Peter Henderson
University of Southampton, Southampton, UK
Chris Rostron
ICL, Manchester, UK

Implementation of metrics in development of highly-safety critical SW

Kenneth Kvinnesland
Navia Aviation , Oslo Norway

Introduction of Metrics in Civil Engineering Software Development

ESSI PIE No. 27272 - ICENSOM

Dr. A. Tsipianitis
TEGEA SA, Athens, Greece

1. Introduction

In the recent years measurements play a critical role in achieving effective software development. Software measurement techniques are increasingly being used by organisations aiming to improve the quality of the software they develop and the efficiency with which it is produced. This trend can be justified as a systematic approach to overcome the effects of software crisis: poor quality systems delivered late and over-budget. According to various studies, measurements should be introduced in a top-down approach focused on specific goals, which are consistent with the business of the organisation and the software development process in use. Furthermore, the measurement results should be interpreted based on the organisational context and business objectives, and should be used for managing and improving the software development process.

The ICENSOM Process Improvement Experiment (PIE) under the CEU programme ESSI (European Systems and Software Initiative) introduced at TEGEA SA (a civil engineering software developer) a rigorous discipline for software quality measurements driven by business objectives. The rationale behind this PIE is based on the fact that software packages developed by TEGEA need to be frequently updated after changes in Greek and European regulations on construction and environmental engineering. This frequent updating is a major concern because it has to be carried out in a controlled manner, for the achievement of small lead-times, without compromising the quality and reliability of the developed product.

The technology that was introduced to alleviate the above problem is the *Goal-Question-Metric* (GQM) method. GQM was applied in the context of a typical project developing software for static analysis and dimensioning of buildings. Such measurements will enable the identification of appropriate areas for improvement in software development practices and will support the implementation and subsequent monitoring of the improvements.

TEGEA benefits from introducing GQM in determining potential process improvements and in supporting related improvement actions. Furthermore, TEGEA expects a higher product quality and reliability, shorter time-to-market and less software development costs. The potential benefits within the wider European community are also significant because the PIE deals with a common concern and promotes a proven method.

This paper presents the motivation for the ICENSOM PIE, a brief overview of the GQM method, the objectives and the organisation of the experiment, an account of the activities, up to date experiences and lessons learned from ICENSOM PIE and a set of conclusions and plans for improvement activities.

2. Starting Scenario

Development projects at TEGEA's Engineering department involve a number of typical attributes. Such attributes include an effort of 30-60 person-months and a duration of 9-12 months. After the completion of each project, a follow-up period of up to six months is commonly used for collecting feedback from the use of the software product at customer's premises. Typical project teams involve 4 or 5 persons. The project team members are involved in one or more functions such as software design, coding, testing or administration.

TEGEA's software engineering practices are based on the waterfall model. Additionally, practices based on prototyping methods are used for critical software system parts. Efforts are currently invested in formalising and documenting the current practices for software development.

Newly-appointed individuals undergo on-the-job training and are supported by more experienced personnel. Each software project has a nominated Project Manager, that due to the size of the company might occasionally coincide with the General Business Manager. There is no independent Software Quality Assurance (SQA) function at TEGEA, due to its small size. However, some SQA activities are carried out by the project manager and other project technical staff. As far as it concerns software development methods, technologies or tools, no systematic training is currently in place.

The General Business Manager evaluates the feasibility, expected benefits and risks from each software project, before undergoing any contractual commitments. This activity is not performed in a formal and documented way. Project management conducts periodic reviews of the status of each software project at major milestones, but not in a formal way. Project managers carry out estimation and scheduling activities, using their experiences and knowledge of the abilities and availability of the project resources they administer.

No formalised procedure exists to control changes in software requirements, design specifications, accompanying documentation and code. The functionality and quality of the software system under development are reviewed by area specialists. To this respect, all major software products and documents are technically reviewed by project staff. The testing and verification of all implemented functions in the software systems is ensured by test specification documents, which are based on requirements. Additional tests are carried out when major faults are detected during system test activity. Acceptance tests are also performed by end users in certain cases.

Coding standards are usually applied to the software projects. The majority of products are developed using the TURBO PASCAL and C++ programming languages in Windows environment. Software design is documented through flowcharts and appropriate textual descriptions. Project staff makes use of several documentation tools. Software tools such as the MS Project are used for project planning.

Until the ICENSOM PIE there was no rigorous discipline for software measurements. No documented and widely used measures are utilised for the estimation of software product size, and thus productivity. Managers estimate project resources by using empirical methods, based on accumulated experience. Actual project data and estimates on assignment of resources are recorded but not analysed. There is no complete and stable baseline for providing reliable statistics on software code errors or test efficiency. It should be noted that, problem reports by end users are recorded.

The strengths of the current practices at TEGEA include the utilisation of technical and management reviews for software development projects. Personnel has high technical competence on civil engineering software issues. People are willing and capable to work in teams for development and are committed to quality and improvement issues.

The weaknesses of TEGEA's software practises concern the lack of a formal framework for software measurements, whereas specific functions (such as SQA, methods & tools support, change control) are not implemented. Furthermore, software project planning and estimation is carried out empirically. Therefore a number of corrective actions are necessary including: introduction of formal software quality measurements, enhancement of the definition and documentation of software development practices, implementation and enhancement of supporting functions (such as quality assurance, technical reviews, product testing and change control) and improvement of project management practices.

3. Overview of the GQM Approach

There are a variety of approaches for establishing measurement programs that have appeared in the literature. Among the various methods for software measurements [1, 2, 3], the Goal-Question-Metric (GQM) approach [3] is one of the most effective and well-established ones. The GQM method was developed by Professor V. Basili and his research group at the University of Maryland, in close co-operation with NASA Software Engineering Laboratory. Since then, the method has also been applied by several software development organisations, including Ericsson, Daimler-Benz, Bosch, Schlumberger and Nokia, among others. The method is based on a simple process by which software developers and managers first define the goals that the software process and its related products must achieve (on organisation and project levels), then refine the goals into a set of questions and finally identify the metrics that must be provided to be able to answer the questions. Thus, GQM provides a top-down approach to the definition of metrics, whereas the interpretation of the measured data is done in a bottom-up way. This helps software developers and managers to share a common view of the target of the measurement, knowing both what to measure and for which purpose the measured data will be used.

The result of the application of GQM is the specification and implementation of a measurement plan for a particular set of goals and a set of rules for the interpretation of the measurement data within the context of these goals. The GQM model has three levels:

1. *Conceptual level* (GOAL): A goal is defined for an object (product, process, project or resource), for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment.

2. *Operational level* (QUESTION): A set of questions is used to characterise the way the assessment / achievement of a specific goal will be performed based on some characterising model. Questions try to characterise the object of measurement (product, process, etc.) with respect to a selected quality issue and to determine either this quality issue from a selected viewpoint or the factors that may affect this quality issue.

3. *Quantitative level* (METRIC): A set of data is associated with every question in order to answer it in a quantitative way. The data can be objective (e.g. person hours spent on a task) or subjective (level of user satisfaction).

A GQM model has an hierarchical structure starting with a goal, that specifies the purpose of measurement, the object to be measured and viewpoint from which the measure is taken. The goal is refined in several questions, that usually break down the issue into its major components. Each question is then refined into metrics. The same metric can be used in order to answer different questions under the same goal. Several GQM goals can also have questions and metrics in common, provided that when the measure is actually collected, the different viewpoints are taken into account correctly (i.e. the metric might have different values if taken from different viewpoints).

With the GQM method, the number of metrics that need to be collected is focused on those that correspond to the most important goals. Thus, data collection and analysis costs are limited to the metrics which give the best return. On the other hand, the emphasis on goals and business objectives establishes a clear link to strategic business decisions and helps in the acceptance of measurements by managers, team leaders and engineers.

The GQM approach can be used as stand alone for defining a measurement program or, better, within the context of a more general approach to software process improvement. A good approach for software process improvement, that is compatible with the GQM approach, is the Software Engineering Institute's Capability Maturity Model (CMM) [4] combined with Deming's widely used Plan-Do-Check-Act cycle for improvements implementation. Another approach is the Quality Improvement Paradigm (QIP) an iterative, goal-driven framework for continuous improvement of the software development [5]. This method is actually an offspring from the development of the GQM one. Because information necessary for applying the GQM method is derived and/or used in every step of QIP, GQM has also been described as the measurement view of the QIP.

The GQM approach to measurement of processes and products has been used successfully in selected industrial environments within the CEMP ('Customised Establishment of Measurement Programs') project. This project was funded by CEU within the ESSI framework and aimed at evaluating the GQM approach and supporting its transfer into industrial software engineering practices [6].

The average cost resulting from the measurement activities on a project using the GQM approach is around 5% of the total cost for the software development. This additional cost is much less when compared to the cost of bottom-up metrics approaches, which are based on collection and analysis of large amounts of data.

References

[1] K. H. Moeuller and D. J. Paulish, "Software Metrics: A Practitioner's Guide to Improved Product Development, Chapman & Hall, 1992.

[2] R. B. Grady, D. L. Caswell, "Software Metrics: Establishing a Company-wide Program", Prentice Hall , 1987, ISBN 0-13-821844-7.

[3] V. R. Basili, G. Caldiera, H. D. Rombach, 'The Goal Question Metric Approach', *Encyclopedia of Software Engineering,* volume 1, John Wiley & Sons, 1994, pp. 528-532

[4] M. C. Paulk, C. V. Weber, B. Curtis, M. B. Chrissis, "The Capability Maturity Model: Guidelines for Improving the Software Process", Addison-Wesley Publishing Company, 1995, ISBN 0-201-54664-7.

[5] V. R. Basili, G. Caldiera, H. D. Rombach, 'Experience Factory', *Encyclopedia of Software Engineering,* volume 1, John Wiley & Sons, 1994, pp. 469-476

[6] CEMP ESSI Project #10358, Final Report version 3.0, 1996, VASIE - ESSI PIEs Repository (http://www.esi.es/VASIE/).

4. Objectives and Organisation of the Experiment

The ICENSOM experiment involves the following objectives:

Technical Objectives of the Experiment

- introduction at TEGEA, of formal software quality measurements, based on the GQM method
- determining potential process improvements at TEGEA and introduction of activities for supporting the related improvement actions
- assessment of the GQM method, concerning its suitability for engineering SMEs
- dissemination of experiment related experiences and lessons learned, towards appropriate audiences in Greece and the rest of Europe

Commercial Objectives of the Experiment

- higher product quality and reliability as experienced by the customer
- less time-to-market for TEGEA's software products
- less software development costs due to less re-design, testing and maintenance activities

The anticipated technical and commercial benefits from the ICENSOM experiment include: establishment of potential improvements, introduction of a mechanism for supporting the corresponding improvement actions and enhancement of project management practices. Moreover, the ICENSOM experiment will result in higher quality in the developed software products, increased efficiency of technical review and testing activities and enhanced competence and motivation of TEGEA's staff. Finally, ICENSOM will contribute towards transferring of experiences from this PIE to other activities in TEGEA, less time-to-market for TEGEA's software products, reduction of the software development cost and higher productivity for software development activities.

The experiment involves three main phases: the Experiment Set-up, the GQM Application and the Experiment Conclusion. The *Experiment Set-up Phase* involved resolving any synchronisation issues with the baseline project, training of the involved personnel, establishment of a 'Measurements Responsible' role in TEGEA's Engineering department and setting-up the co-operation with the subcontractor. The deliverables from this phase include the Training Plan, the Training Reports and the Subcontractor Work Specification.

The *GQM Application Phase* consists of the four following main activities: Primary Goal Derivation, Goal Decomposition, Metrication and Measurements Interpretation.

Primary Goal Derivation: During this activity the current software development practices at TEGEA were analysed with emphasis on the baseline project. This analysis was performed based on appropriate questionnaires and interviews, taking also into account any existing information from analogous past activities. The outcome of this analysis was the Assessment Report, including findings, remarks and directions for potential improvements. Then definition of the primary goals that will be used later on for measurements definition was carried out. This activity was based on information from past projects, the results of the analysis of software development practices at TEGEA and the special constraints of TEGEA's development environment. The primary goals definition was organised by the Measurements Responsible, supported by the external consultant, and involved personnel from the baseline project and other projects at TEGEA's Engineering department. The outcome of this activity was the Goal Analysis Report.

Goal Decomposition: The previously derived primary goals were decomposed into sub-goals, questions and the corresponding metrics. This activity resulted in a Goals Tree incorporating questions that led from goals and sub-goals to the metrics corresponding to these goals. The Goals Tree was also included in the Goal Analysis Report.

Metrication: This activity involved preparation and planning for the collection, processing and subsequent analysis of measurement data. The GQM Measurement Plan was prepared and introduced (including complete measurement definitions) in the activities of the baseline project. This plan was prepared by the Measurements Responsible and was appropriately reviewed by individuals form the baseline project. Then, primitive data were collected from the baseline project, and after appropriate verification and processing led to measurement results.

Measurements Interpretation: During this activity the defined measurements in the GQM Measurement Plan are validated against the goals and questions. Subsequently, analysis and review of validated measurement data is performed in order to assess the degree of achievement of the defined goals. As a result, the quality level of the products and processes being measured is evaluated. This analysis of validated measurement data leads to the identification of potential process improvements and enables monitoring of improvement actions. The associated deliverables include the Measurements Analysis Report and the Improvements Report.

During the *Experiment Conclusion Phase*, the degree of suitability and effectiveness of GQM in the context of TEGEA is established and experiences/lessons learned from the ICENSOM PIE are disseminated towards various Greek and European organisations developing software. Deliverables from this phase involve the Experiment Evaluation Report and the Post-experiment Activities Plan.

The *baseline project* for the ICENSOM experiment is called SKAT-II and involves the development of a user friendly and reasonably-priced software package for static analysis and dimensioning of buildings completely made out of pre-constructed steel concrete elements. This construction method, named 3D-Method (Three Dimensional Structural Wire-Mesh with Embedded Insulating Material), is particularly suitable for timely and cost effective building constructions. The software package to be developed will incorporate the following features:

- static analysis and dimensioning of building foundations, steel concrete wall elements and plates
- user interface based on mouse-driven commands and pull-down menus
- interface with spreadsheets, such as EXCEL, for estimating the cost of the constructions under analysis
- interface with CAD packages like AUTOCAD, for 3D representation of the load carrying elements of the building under analysis

This project is of major importance to TEGEA due to its innovative character and its high market potential. The duration of the baseline project is 10 months and involves an effort of 55 person-months.

5. Conducting the Experiment

The analysis of the software development practices in use at TEGEA's Engineering department was carried out as a first step in applying the GQM method. The requirements and practices at Levels 2 and 3 of the Capability Maturity Model (CMM) were used in conducting this analysis. The CMM was chosen as a framework for this analysis because it is widely recognised by many industrial organisations, provides a reliable picture and is compatible with the GQM approach. The analysis was performed using appropriate questionnaires and lasted for five days. It focused on the baseline project SKAT-II but relevant information and issues from other TEGEA's software projects and previous analysis findings were taken into account. The results of the analysis of the software practices will support the activities for planning and organising process improvement at TEGEA. These results are presented in the Assessment Report.

The remarks and findings from the analysis activity were checked with baseline project individuals and were prioritised. In this manner, focus is directed towards the most important issues for measurement and subsequent improvement, where positive impact is possible. In the Assessment Report recommendations were also included for each finding to support the preparation and establishment of future improvement plans. The findings from the analysis of software practices were categorised in the following areas: software project management, software measurements, product control & quality assurance, technical reviews, development processes and training.

In line with the GQM approach, the *Goal Tree* was then established, aiming at an improved software development process and better product quality. The results from the above analysis, information from past projects and particular requirements and constraints of the baseline project were used as input. The Goals Tree contains the primary goals, which are further analysed into sub-goals and corresponding questions that lead from the identified goals to the associated metrics. This structure is documented in the Goal Analysis Report. The Goal Tree was intentionally kept simple and straightforward in order to facilitate the succeeding activities in the GQM method (the definition, collection, validation and analysis of measurements). The Goal Tree is given in Figure 1, and was used as a basis to derive the full Goal Tree with goals, sub-goals, questions and metrics. The Goal Tree is structured in a tabular format, involving the sub-goals, factors and questions that affect each primary goal and subsequently define the relevant measurements.

GOAL 1. Enhance the effectiveness of the software development process

Goal 1.1 Enhance activities for the definition of software development processes

Goal 1.2 Enhance project management activities for software development

Goal 1.3 Establish data for use in estimation of software development projects

Goal 1.4 Improve consistency and precision in achieving agreed delivery dates

Goal 1.5 Reduce redesign effort and cost in software development projects

GOAL 2. Increase control on and quality of the developed software products

Goal 2.1 Monitor the quality of software products in all phases of their life-cycle

Goal 2.2 Improve the effectiveness of software testing activities

Goal 2.3 Increase use of technical reviews for the software products

Goal 2.4 Enhance competence of both technical and management personnel

Figure 1. Goal Tree of ICENSOM experiment.

The application of GQM measurements at TEGEA's Engineering department was centred around two primary goals: enhancing the software development process and increasing the quality level of developed software products. This approach was influenced by the characteristics and constraints of the software development environment at TEGEA, as well as by the related business goals and priorities. Each defined sub-goal in the Goal Tree (fig. 1) was analysed by the ICENSOM team and involved baseline project personnel in order to identify any factors that can impact (either positively or negatively) the fulfilment of the specific sub-goal in concern. The identified factors then led in a set of questions that address the sub-goal in concern and finally, to the related metrics. Each question corresponds to one identified factor. The defined metrics are actually the answers to the identified questions.

An example of identified questions that address a particular sub-goal, namely Goal 1.2, in the Goal Tree presented above is depicted in Figure 2.

Goal 1.2 Enhance project management activities for software development

Question 1.2.1 How to control and enhance planning precision?

Question 1.2.2 Is there a process in use for project planning and tracking?

Question 1.2.3 How to monitor and control periodic project status reviews?

Question 1.2.4 How to monitor and control risk identification and analysis activities?

Figure 2. Example of questions related to a sub-goal

The next activity in applying the GQM method involved identifying metrics for the factors (and questions) associated with the sub-goals in the Goal Tree. Figure 3 depicts an example of metrics for the case of the sub-goal for 'enhancing project management activities' (Goal 1.2 within the primary goal for increasing the effectiveness of the software development process).

Goal 1.2: Enhance project management activities for software development

Factor: Risk handling activities in software development projects

Question 1.2.4: How to monitor and control risk identification and analysis activities?

Metrics:

1. No. of project meetings for risk handling (identification and analysis)
2. Effort spent in project meetings for risk handling (no. of hours for such meetings and percentage of these hours in the total project effort)
3. No. or risks identified in project meetings for risk handling

Figure 3. Example of metrics associated to a question

The activities for identifying metrics for the baseline project involved rather simple techniques (group reviews and brainstorming) and reflected the actual practices used at TEGEA for software development and management, as well as the experiences of the involved individuals. The approach to keep things as simple as possible and to involve all interested roles and individuals was followed throughout the ICENSOM experiment. Such an attitude is in fact a prerequisite for ensuring acceptance, participation and commitment of the development team towards improvement goals and activities. Using this approach, simple metrics were defined and basic statistics or graphs were used for the analysis and presentation of the measurement results.

The GQM measurement plan for the baseline project was then prepared, including all defined measurements and all the necessary information for the collection, analysis and validation of the defined measurements. This plan provides for each defined measurement information that indicates: who collects the data, when the data are collected, how measurement data are processed, validated and presented. The measurement plan was implemented in the baseline project and sets of measurement data were collected, validated and analysed. These activities are carried out by the ICENSOM team in co-operation with baseline project personnel as necessary.

6. Results and Experiences

Software Process Improvement: The GQM method can provide a suitable framework for measurements that facilitate improvements in the software development process. This approach can provide a way to prioritise identified improvement areas. In this respect, GQM supports definition of project goals, analysis of measurement data and feedback into the project and organisation. Furthermore, it allows the definition of measurement plans that can be reused in later projects.

GQM Goals and Measurements: Prior to the application of GQM method, there was very limited use of measurements at TEGEA's software projects. Therefore, the identified GQM goals for the baseline project (SKAT-II) were focused in consolidating measurement data and establishing a baseline for understanding the current state of software development practices. On the other hand, GQM goals for improvement of software development practices and products were also identified. The measurements previously used at TEGEA did not provide significant help in supporting project or business goals and improvements. Furthermore, in defining such measurements constraints and views of project managers or developers were not taken into consideration. The GQM method on the other hand depends on the consolidation of ideas and concerns of various roles, both technical and management.

Thus, the GQM approach contributes in increasing motivation and approval of the measurements framework by all individuals involved in a project. A crucial factor in establishing a successful measurement framework is to incorporate, during the Goal Derivation and Decomposition activities of the GQM method the existing business goals, already used measurements and specific constraints of the organisation. Thus, consistency of GQM measurements with existing practices is facilitated and commitment of personnel to the new measurements is ensured to a great extent.

GQM Data: For projects and units with limited experience and use of measurements, special care should be put on collection, analysis and validation of the measurement data. Data collection must be carefully planned (in terms of responsibilities, time period and supporting templates or forms) since this ensures the correctness of the measurement data. Analysis and interpretation of the measurement data is the hardest part of measurement. This should be carried out in co-operation with all involved individuals in a team-based manner. For example, it is dangerous to provide results that address (even indirectly) the effectiveness of software developers or the quality of software work products, especially in cases were such results are given as granted.

Application of GQM: During the application of a new method in an organisation, one should try to keep things simple and take a more conservative approach. In this respect, the most important and necessary goals and metrics should be implemented in order to keep cost at a reasonable level, especially for a small organisation. It is very tempting, especially in less mature projects or organisations, to measure everything for the sake of completeness and therefore result in large sets of data that is very difficult to handle.
The use of the GQM method at TEGEA contributed to a great extent in the understanding of the various activities and phases of software development. The importance of a adequately defined software process was recognised and accepted by TEGEA's personnel, both managers and developers, involved in ICENSOM experiment.

Overhead of GQM: The effort spent up to date in the ICENSOM experiment amounts to a significant overhead for software development activities. The GQM introduction corresponds to an overhead in the order of 20% compared to the effort spent in the baseline project. Although this overhead is large for an SME organisation, it is expected that future applications of GQM method to development projects will involve a greatly reduced overhead. This is due to the already acquired skills for the GQM method, as well as to already established roles and infrastructure.

Organisational support: A prerequisite for the successful introduction and implementation of the GQM framework in an organisation is to make it part of the company strategy and business goals. The management of the organisation should have the vision of process improvement and should provide the necessary resources and support. The results from the introduction of the GQM method through ICENSOM will be used, after appropriate analysis and tailoring in next software projects. Acceptance of future applications of the GQM method is ensured through the internal dissemination activities of ICENSOM and the involvement of key personnel from TEGEA in the GQM introduction.

Positive Aspects of the Experiment

The *positive aspects* of ICENSOM experiment are listed as follows:

ICENSOM team and involved personnel form the baseline project has acquired skills with respect to the GQM method and enhanced their awareness for process issues.

Support of the external consultant significantly facilitates ICENSOM experiment. The training and consulting activities for the PIE provided valuable guidance, experience and various viewpoints in the derivation and analysis of the GQM goals and measurements.

The GQM method provides a suitable framework for measurements that facilitate process improvements. GQM involves the concerns and viewpoints of both managers and technical staff in the organisation, thus enhancing their motivation for and commitment to measurement and improvement activities. Moreover, the involvement in the GQM activities promotes the team-based approach in carrying out project and organisational activities. The GQM method actively involves all interested individuals from the very beginning (from goal derivation until measurements analysis and validation).

Close co-operation between the baseline project personnel and the ICENSOM team is essential for introducing successfully the GQM measurements at TEGEA.

Negative Aspects of the Experiment

The *negative aspects* of ICENSOM experiment are listed as follows:

The application of GQM method to a baseline project should be well planned and performed gradually, in order to minimise interruptions and negative impacts in the software development activities. The introduction of a new method is usually not without problems and time has to be devoted in coping with unexpected difficulties.

Management of the baseline project was concerned about delays and interruptions in the product development schedule due to additional activities imposed by the ICENSOM PIE. It is rather common that the return-on-investment tends to be small or even negative in the first project were a new method is introduced. Such concerns could be rectified by the expectation of long-term benefits, evident in future projects.

During goal derivation of GQM, certain goals initially reflected short-term concerns for the baseline project. These goals were influenced by immediate priorities of the baseline project manager and technical personnel. However, such goals undermine the effective use of measurements that can facilitate long-term process improvements.

It takes rather long time between the definition of GQM measurements and obtaining actual results. In this respect, it is not possible to obtain all measurement results from the baseline project within the time frame of the ICENSOM experiment. Furthermore, the derived improvements during the ICENSOM experiment will be evident after the end of the PIE.

In case of inadequate previous use of measurements, as with the baseline project, it is difficult to evaluate all the identified goals and improvement actions. This is due to the lack of a current baseline to make comparisons with the achieved results.

7. Conclusions and Further Activities

The ICENSOM experiment has been very helpful so far in introducing the GQM method at TEGEA. The GQM method provides a suitable framework for measurements that facilitate improvements in the software development process. A significant interest has been expressed at TEGEA to continue the GQM measurements to future projects and investigate the expansion of the measurement program. The decision for wider application of GQM will be taken by TEGEA's management after the conclusion of ICENSOM by considering the final results and evaluation.

Certain activities are currently carried out within ICENSOM, while other activities are planned for the period after the conclusion of the experiment. These activities are included in the following list:

The collection, analysis and validation activities of the defined measurements in the baseline project are carried out to completion.

Establishment of the degree of effectiveness of GQM in the context of TEGEA. TEGEA's business environment is being initiated. This activity is based on an overall analysis of results form the baseline project. This evaluation will include any shortcomings, problems and proposed solutions regarding the application of GQM.

Performing the rest of the planned internal presentations of ICENSOM results. Activities of external dissemination have also been scheduled.

Investigation and scheduling of activities for packaging the acquired results and experiences from ICENSOM for further use at TEGEA. The scheduling of introduction of the GQM method in future projects is based on the ICENSOM results and experiences acquired so far.

Investigation will be initiated for the introduction of tools to automate and tailor the GQM method in future development projects. Such tools should be able to incorporate the already acquired experience and results from the first GQM application through ICENSOM.

Appendix A. Author CV

A. Tsipianitis has a diploma in Civil Engineering from National Technical University of Athens (NTUA) and a Ph.D. from University of Hamburg in Geo-sciences. Since 1989 he participated in various research projects of NTUA in the areas of environmental engineering, maintenance and restoration of Greek historical buildings. He carried out several technical studies in the areas of environmental engineering, geological studies, port construction, undersea piping construction and construction of special purpose buildings (depots, schools etc.). Dr. A. Tsipianitis has also carried out the evaluation and management of significant construction projects in the Greek public sector funded by the CEU.

Dr. A. Tsipianitis is the Business Manager of TEGEA S.A., an engineering firm active in the areas of environmental engineering, construction projects and development of software packages for construction and civil engineering.

For correspondence:

Dr. A. Tsipianitis,
TEGEA SA, 100 Alexandras Ave., Athens 114 72, Greece
Tel.: +301 6440103
Fax.: +301 6424001
E-mail : atsipi@tee.gr

Appendix B. Company Description

TEGEA is an engineering firm established in 1991, active in the following areas:

- technical analysis, design and implementation of construction projects (buildings, ports, roads, water distribution and sewage networks)
- technical analysis, design and implementation of environmental engineering projects
- provision of services / consulting for the management of construction works and environmental engineering projects
- Development of software packages for computer aided analysis and design of special constructions. These packages target the market of civil engineering firms, as well as the market of building construction companies.
- Development of software packages supporting the technical and financial management of large construction projects. These packages target the market of construction engineering companies.

Most developed software packages are also used internally, in the context of TEGEA’s construction and consulting contracts.

SIMMER: Software and Systems Integration Modelling Metrics and Risks (Getting to Level 4)

Brian Chatters
ICL, Manchester, UK

Peter Henderson
University of Southampton, Southampton, UK

Chris Rostron
ICL, Manchester, UK

Introduction

This article documents the mid-term progress and provisional conclusions of SIMMER; an ESSI funded Process Improvement Experiment. The overall objective of the experiment is to produce a more effective means of planning and controlling complex software and systems integration projects.

In order to remain competitive, ICL (as well as many other companies) needs to continually improve its predictability of costs and schedules for integration projects, to reduce time to market and to reduce costs without detriment to the quality of the products. The developments of our complex software and systems rely more and more on using commodity components and collaborations as a way to meet these business objectives. The ability to accurately predict effort and time scales and the ability to keep within budget is becoming increasingly difficult in such projects.
The specific purposes of the experiment are to demonstrate the applicability of the "Cellular Manufacturing Process Model" (CMPM) technology to a business critical, live software and systems development project and to develop and tailor the model and associated metrics to improve the project management processes.

Starting Scenario

Over recent years, software and systems development has become more complex and the trend has been towards the use of bought-in components. There is an expectation that, by buying in components, the time to bring a system to market can be significantly reduced. However, the use of third party components introduces a number of unknowns into the development activities, increases risks and jeopardises delivered quality. This fact is particularly true when the component is software that may not have been exposed to the specific operational environment previously, often leading to performance problems.
An internal assessment of our development processes (incidentally, achieved by participation in the SPICE/2 trial) identified the need to improve:

- the supplier management process - to better integrate the supplier's engineering processes with those of ICL, particularly in the area of support during software and systems integration
- implementation of organisational wide process metrics to help understanding and improve predictability

SIMMER makes a significant contribution to addressing these two areas for improvement. Traditionally, ICL has used the V-diagram waterfall life cycle model to plan and control software and systems development. Estimates of effort and time scales are based on an understanding of the architecture of the solution, and expert opinion of the degree of difficulty and potential problems likely to be encountered during the integration activities. The availability of resources and the team size is also taken into account when making the estimates.
A number of alternative life cycle models and development methods, such as DSDM [7] and Boehm's spiral model [2], are now in existence but none of them adequately address the issue of managing complex integration of third-party supplied components.
The CMPM, developed jointly by ICL and Peter Henderson of Southampton University, is a more appropriate model for the changing

business. It provides a better means of capture and metrication of the interfaces between the contributing supplier, development and integration activities, enabling earlier and more comprehensive verification and validation of software products.

The Cellular Manufacturing Process Model

General Description

The CMPM is defined to be a set of "Manufacturing Cells" with the relationships between the cells described as a set of metrics. The model is based on Watts Humphrey's network models of software development [4], and on the "value chain" model developed by Michael Porter [5].

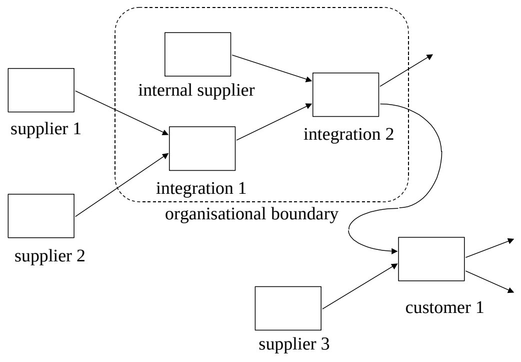


Fig.SIMMER.1: The Cellular Manufacturing Process

The model is based on a view of products that are integrated from a mixture of bought-in and self-built components (Fig.SIMMER.1). In this context, system integration is defined to be those activities which identify (and specify) components and develop "glue" to bind them. Some components will plug directly in (that is, they will not require any additional glue). For such components, the choice of one influences or restricts the choice of the others. The nature and quantity of glue required is a significant property of the system design. Each integration activity is defined as a cell within the CMPM. The model is clearly hierarchical. Each cell can be a component in a higher-level integration activity. Each component used by a cell can, itself, have been integrated from lower level components, either by in-house development or supplied by a third party.

Products with hierarchical structures lend themselves to being developed and built in a network of manufacturing cells. Each cell is responsible for one level of integration. The cell receives components from suppliers, makes some components locally, glues the components into an integrated product (which is tested to output standards) and ships the integrated product to a customer or to the cell performing the next level of

integration. Note that the traditional software development process can be defined as an integration cell within the definition of the CMPM. It makes all components (lines of code) in-house and glues (compiles and builds) them into a software module or software product.
The behaviour within a cell can be as formalised or as ad hoc as the business or product demands. The measurement regime of the CMPM (the metrics) is not dependent upon detailed knowledge of how each cell performs its integration tasks, only how it meets its external obligations.

Metrics within CMPM

Six metrics are defined by CMPM (Table 1). Fig.SIMMER.2 illustrates how these metrics are associated with each cell within the network. The reassuring thing about these metrics are that they are clearly at the management level, not down at the detailed code level.

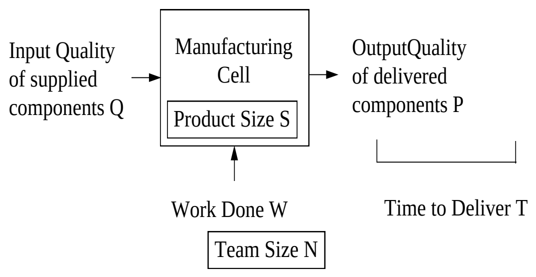


Fig.SIMMER.2: Metrics Associated with Each Cell within CMPM

The measurements of the quality of the input to and output from each cell are converted to percentage reliability measures (100% indicating total reliability). The values will be based on actual measures of the quality or based on an expert judgement from the project team members. Clearly, the quality metrics are directly related to project risks.

W	Effort	*The work done on each cell, in net person days (excludes project management overheads). Factors affecting W include unforeseen problems, changes in requirements, poor estimates of S, late handovers, and product problems (the number of and the cost of resolution that may be further complicated by supplier support interfaces).*
T	Elapsed time	*The schedule of deliveries of components from each cell, in elapsed working days (excludes weekends and public holidays).*
N	Team size	*The average team size (W=T*N)*
S	Size	*A measure of the size of the product, in "Standard Integration Units" (SIU's) - in the context of integration, size is determined by "hard" cost drivers which are quality independent. The drivers cover costs for building systems, installing products, regression testing, producing project infrastructure, and making glue/in-house components.*
Q	Input quality	*The average quality of incoming components, on a scale of 0 to 5 - the costs incurred as a result of the values assigned to this metric (and the required output quality P) are determined by "soft" drivers which are dependent upon the issues, risks, and problems inherent in the supplied components.*
P	Delivered quality	*The target quality of outgoing components, on a scale of 0 to 5*

Table 1: CMPM Metrics

Predicting Costs and Time Scales Using CMPM

COCOMO (Constructive Cost Model) is a model for estimating software effort, cost and schedule for a number of different types of software development projects [1]. COCOMO predicts a relationship between S, W, N and T. CMPM postulates that a relationship also exists between Q, W and P. The conjecture is that effort can be modelled against size (S) but that input and output quality (P and Q) will also have a significant impact on any predicted costs. That is:

$$W = f\,(Q,P,S)$$

Where W = effort, Q = input quality, P = output quality and S = size.

The function will demonstrate the behaviour that, if the target P increases, either W will increase or Q needs to be increased. Similarly, if S increases, then W or Q will need to be increased in order to achieve the same level of output quality (Fig.SIMMER.3).

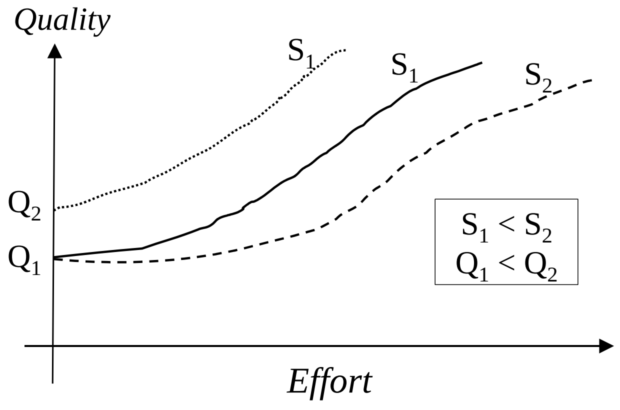


Fig.SIMMER.3: Illustration of effect of S, P and Q on W

The function can be determined from historical data and used to predict costs and time scales for future projects.

Relationship of CMPM to Capability Maturity

The effectiveness of CMPM as a predictor of costs will depend upon how well an organisation understands its processes. Only then can it make reasoned estimates of the size of the tasks that need to be performed to achieve its deliverables. There are (at least) two key factors that affect how well an organisation can make predictions.

Firstly, it needs to ensure that its activities are **managed** effectively. To achieve effective management, it needs to gather data on the performance of its activities and to analyse the data to identify potential predictors of future performance. Basic data collection of cost, schedule and problems is a requirement of level 2 of SPICE [6].

Secondly, it needs to be able to define the tasks that need to be performed within a cell. The CMPM allows the description of a cell to be as formal or as ad hoc as the business demands. However, to gain maximum benefit, our experiment demands a level of description which enables project manager's to plan detailed tasks in a repeatable and, to some degree, a predictable manner. To achieve this, we have advocated that the project's processes need to be **defined**. That is, they need to operate at level 3 within the SPICE model before the application of CMPM can have full effect.

With these two key building blocks in place, CMPM can then be used to further improve the predictability of the performance of a project (hence, the reason for the subtitle of this article: "Getting to Level Four").

Plans and Expected Outcome

Objectives

The overall business objective of this experiment is to produce a more effective means for the design and planning of complex software and systems integration projects, involving the use of commodity components, collaborations with third parties, or the reuse of existing components. The new process will improve the understanding of how to exploit the components in a new project and hence, help to mitigate against risks and to improve predictability.

Specifically, the aims of the experiment are:

- to demonstrate the applicability of the CMPM technology to a business critical, live software and systems development project;
- to develop and tailor the model specifically for software and systems integration and validation;
- to develop appropriate metrics to support the project;
- to quantify the benefits as a result of applying the technology.

Benefits

The specific benefits are expected to be:

- **More accurate predictions of costs and schedules.** The major expected benefit by the application of the new method will be to significantly improve the accuracy of the predictions of project costs and schedules, thus enabling more realistic forecasts. This improvement, in turn, will lead to better overall business planning and enable the organisation to have more confidence in ensuring that its return on investment will be protected.
- **Improved Product Quality.** The new method will ensure that a better focus is given to quality requirements and ensure that preventive action is planned to mitigate against potentially high-risk components. This focus, in turn, will ensure better test and validation coverage. The process improvement will have the effect of significantly reducing the number of customer detected bugs once the product is released, resulting in a reduction in support and maintenance costs and improved customer satisfaction.
- **Reduced Time to Market.** The biggest cause of delays to the

planned schedule of a project involving supplied components, is the time it takes to resolve unexpected problems. A better understanding of the quality of the input and its impact on the project will enable such problems to be pre-empted and resolved much faster, thus reducing the time to market.

- **Exploitation of CMPM.** The results of the experiment will allow the CMPM to be enhanced and exploited as a key aid to improving software and systems supply, development, and integration processes. The method will be able to be applied to any organisation, which designs, integrates or tests complex software systems, using commodity products, collaborating with third parties, or reusing existing designs and components.

Definition of the Baseline Project

The baseline project is part of a broader programme aimed at providing platforms, which exploit emerging technologies and meet the future needs of ICL's customer base. A number of systems management products are included in the system and a minimal amount of non-invasive integration is undertaken to make them easier to use and to improve their RAS (reliability, availability, serviceability) characteristics.

The project is split into seven teams with an average size of eight people. The software is a combination of COTS products, in-house development, and collaborative development with partners. Software products, for example for backup, performance monitoring, printing and event management, are included. The suppliers of the relevant hardware modules provide platform specific software products (e.g. peripheral drivers). Regular, incremental, deliveries of the system are made to the customer base.

The Plan

1. A set of baseline measures will be established by analysing historical data. Variations between predicted and actual measures will be recorded. A log of problem reports from the baseline project activities will be set up, identifying where the problem was found, its severity, which component, process, or supplier caused the problem and how long it took to resolve it.

2. The generic CMPM will be used to define a specific model, tailored to the baseline project. The baseline project activities will be split into cells, which reflect the interfaces between the development teams for each incremental release of the system. A set of metrics will be defined and collected for each cell and initial estimates will be made of the values for these metrics. The input quality levels (Q) of the supplied components will be determined by review and discussion with the project staff and the output levels (P) will be defined by the project requirements.

3. The incurred effort (W), team sizes (N) and time scales (T) of the baseline project will be monitored as part of the normal project

management activities. Every month, actual measures or revised estimates will be made of size (S), input quality (Q) and delivered quality (P), causing revised estimates of total effort to be made, if necessary. Additional data will be collected from the problem management activities and the problem log will be updated.

4. The metrics data collected by the baseline project will be used to carry out ongoing analyses of the relationships between effort (W), size (S), and quality (Q and P). Results of the analyses will be used to evolve a cost estimation model, which will be piloted by the baseline project throughout the lifetime of the experiment to predict effort and time scales for future activities.

5. Data collected throughout the experiment will be analysed and the differences between the predicted and actual values, when compared to the measurements taken prior to the experiment, will be used as a measure of the effectiveness of the process.

Progress against the Plan

Establishment of Baseline Measurements

Effort is recorded and reported each month by project members and gives a breakdown for each incremental release. This breakdown was not available for the historical data and so each team made an estimate on the contribution made to each incremental release.
Representatives from each of the teams also estimated values of S, P and Q. Q was estimated by using a checklist to identify the drivers for and potential causes of poor input quality (Table 2). The representatives were asked to give a rating for each of the suppliers to the cell, on a scale of zero to five, of the degree to which they agreed with the statement concerning the attribute (zero = totally disagree; five = fully agree). An overall rating was derived. This rating was left to the judgement of the team representative because not all suppliers have the same impact on the project plans. However, in many cases, an average was computed. Output quality was also estimated in a similar way, based on hindsight of problems experienced during the implementation of the cell.

Attribute
A. The impact of the product and development process characteristics are fully understood
B. The inherent quality of component will guarantee no problems for your activities
C. The supplier is easy to work with
D. We have full synergy with the supplier
E. The supplier is dependable

Table 2: Attributes used to measure Q and P

Estimation of size S was based on a "standard integration unit" (SIU). All integration projects need to carry out standard tasks such as establishment of the project environment, building systems, installing products, and regression testing. The total amount of effort is also dependent upon the number of supplied components, the amount of in-house development, and the number of incremental builds. The costs for these basic activities are independent of the input quality of the supplied components and the output quality of the integrated system. As an initial attempt to set values of size for each cell, one specific (arbitrary) integration cell was defined to be of size 100 SIU's. Relative values of size were estimated for all the others cells through a facilitated session with all the team leaders and the overall project manager. This approach ensured consistency in the way that size was estimated.

Application of CMPM to the Baseline Project

The project is structured as a network of cells which are classified as “software development”, “hardware development”, “systems integration”, or “build and release” to reflect the different nature of the activities that are carried out within each cell. At this point, it is unclear whether the different categories of cell will display different behaviours, which may need to be reflected in the cost estimation algorithm. Cells are defined for the contribution each team makes to each incremental delivery of the system. Not all teams contributed to every release and, in some teams, activities were carried out in parallel to support a number of releases.

Data Collection

Standard project control processes demand regular progress reports identifying the percentage of activities achieved, the status of the key milestones, issues, and spend to date. The process has been enhanced to ensure predicted and actual values of the six CMPM metrics are reported as well. For each incremental release, each team (cell) within the project estimates values of S, Q and P. The values are then used to estimate W and T from a given N. Typically, a release date is set as a requirement and resources (N) are made available to underpin the milestone, subject to budgetary constraints. As part of the regular reporting, actual or re-estimated values of S, Q and P are recorded and used to review the estimates of the outstanding effort needed to complete the release.

Development of Cost Estimation Algorithms

Early experimentation of the model using the historical data highlighted some inconsistencies in the way that Q and P had been estimated. For example, one team had estimated the same values of S, P and Q for three increments even though the actual effort varied widely. The assumption of the model is that S, P and Q are sufficient synthetics to enable effort to be estimated. If this was the case, equal values of S, P and Q must yield the same estimates of effort. As a consequence, a more objective means of measuring Q and P is required and revised definitions of Q and P are being implemented.
For the development and integration cells,

- Q=5 is defined to be "all components are fault free"
- Q=0 is defined to be "all components are faulty"

A component is defined to be the lowest level unit that, if faulty, will have only one fault. This revised definition requires a value to be set on the

number of components and, for practical purposes, the value is set by estimating the maximum (realistic) number of expected faults. That is, the estimator is asked to consider the worst case scenario which, typically he would do as part of the risk assessment process. For example, a software module may be estimated to have, in the worst case, a hundred faults. Thus the number of "components" in the product is defined to be one hundred.
For the build and release cells,

- Q=5 is defined to be "only one component build is required, per component" (typically, components will be batched together to reduce the total number of builds required)
- Q=0 is defined to be "every component is faulty and requires a rebuild to correct the fault"

In the context of integration, a fault equates to any problem that incurs cost to correct and thus includes process faults, product faults, and issues arising from dealing with suppliers.
P is defined to be the level of achievement against predefined release criteria. If all criteria are fully satisfied, P=5. The project release processes include a formal review of achievement against the release criteria, thus enabling a value of P to be assigned.

Measured Results and Lessons Learned

Baseline Measurements

Table 3 provides an example of the historical data for one team (cell). Each record gives the values of the metrics for the work done to deliver components to each incremental release of the system. The values of Q and P are the subjective values obtained by interviewing team members. As already stated, the plan is to introduce more objective measures of P and Q based on the number of problems expected/experienced.

Project	***W***	***T***	***N***	***S***	***Q***	***P***	***Problem***
software team a	548.5	86	7.6	50	3.1	4	11
software team a	31.92	21	1.8	10	3.6	4	0
software team a	723.1	169	4.8	40	3.6	4	23

Table 3: Baseline Data

Table 4 summarises the actual and estimated values for W, T and N for each incremental release of the system.

	Actual			*Forecast*				
Release	**W_a**	**T_a**	**N_a**	**W_e**	**T_e**	**N_e**	**$(W_a/W_e)\%$**	**$(T_a/T_e)\%$**
release a	2284	149	28	1320	60	28	173	248
release b	440	84	7.9	576	40	18	76	210
release c	2758	213	15	1512	140	14	183	152
release d	793.4	213	4.2	640	140	4.5	124	152

Table4: Summary of Metrics by Release (Actual vs. Forecast)

Much of the extra cost is incurred due to quality problems. The summary shows slips of between 52% and 148% and overspends of between -24% (underspend) and 83%. Note that this data is used for internal planning purposes and does not reflect commitments made to customers. The data provides a starting point against which the results of the experiment can be compared.

Experiments with the Model

The collection of the historical data is providing a better insight into the behaviour of the software and systems integration process. Various hypotheses have been postulated and experiments have been undertaken to model the behaviour using curve-fitting techniques. Fig.SIMMER.4 is an example of the results of one such experiment. It shows the distribution of the values of two constants (b and k) used in a COCOMO-type formula to predict effort from the historical values of S, Q and P.

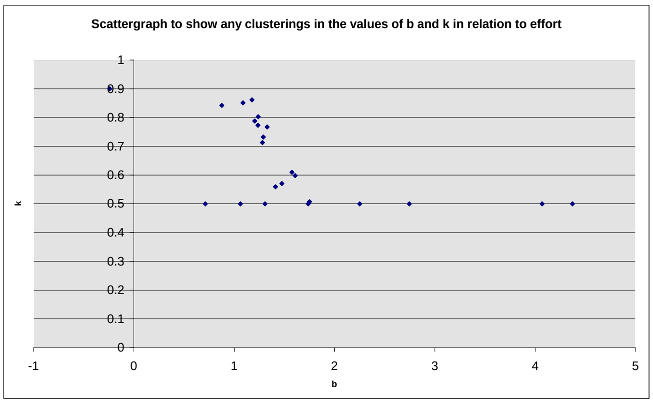


Fig.SIMMER.4: Computed values of b and k

Some of the reasons for the scatter have already been discussed but these early results are encouraging. They suggest that S, Q and P do indeed influence W and a relationship probably exists that can be modelled and used to predict future project behaviour.
Future integration teams will be multi-disciplined and thus, the CMPM may reduce to two cells - one for integration and one for build and release. This change in organisation should help to reduce some of the anomalies, as all teams will perform similar activities. However, the size of the teams will be of the order of existing teams (about 6-8 people) and the difficulties in estimating costs for small teams need to be addressed.

Lessons Learned

Even if we are unable to find values to support the theoretical model, there is still considerable benefit (which should not be underestimated!) to the project in using the metrics set to manage its business more effectively. The metrics will support the management practices that are required to achieve level 4 on the SPICE maturity scale.
Specific lessons learned so far include:

- The experiment only works with the full collaboration from the project.
- The metrics need to be simple and easy to collect. Data collection needs to be established as part of normal business; carried out by project staff. Build on what projects currently care about and are

likely to have data. Then get disciplines in place to capture and analyse the data.
- Metrics provide objectivity into the project management decisions. The act of measuring alone can bring about improvement.
- Subjective measures are better than no measures at all but they are of limited use. They allow projects to think about the issues but they do not allow the development of cost estimation models.
- There is clear benefit in using the extended metrics set (S, P, and Q) in managing integration projects. The issues to be managed in an integration project are much broader than product problems (bugs) and all problems that impact costs significantly need to be considered. Many process problems can have a bigger impact on project overruns (for example, when working with suppliers).
- Metrics need to be based on a sound, objective basis - therefore, a project needs the equivalent of SPICE level 2/3 management practices in place before the full benefits of SIMMER can be realised.

References

[1] Boehm B. et al, The COCOMO II Model Definition Manual, University of Southern California, USA, 1996

[2] Boehm B.W., A Spiral Model of Software Development and Enhancement, in: *IEEE Computing 21(5),* pp. 61-72, 1988

[3] Chatters B.W., Henderson P., Rostron C.J., The Cellular Manufacturing Process Model: Planning a Complex Software And Systems Integration Project, in: *Proceedings of the European Software Measurement Conference,* pp. 559-564, Technologisch Instituut vzw, 1998

[4] Humphrey W.S., Managing the Software Process, Addison-Wesley, 1990

[5] Porter M.E., Competitive Advantage: Creating and Sustaining Superior Performance, The Free Press, New York, 1985

[6] SPICE - PDTR ISO 15504, Software Process Improvement, Part 2: A Reference Model for Processes and Process Capability, Version 2.0, 1996

[7] Stapleton J., DSDM: Dynamic Systems Development Method, Addison-Wesley, UK, 1997

Appendix 1: Author Profiles

Brian Chatters

Brian Chatters is a Software and Systems Engineering consultant within ICL High Performance Systems. He is responsible for ensuring continuous development of the organisation’s software and systems engineering capability and he has developed and introduced a framework based on SPICE and CMM, to discharge this responsibility. He has worked in the software industry for over 30 years in various roles including programming, strategic design and project management. In the last 12 years, he has focused on quality management and process improvement. He graduated from Bristol University with an honours degree in mathematics. He is a fellow of the Institution of Electrical Engineers, a Chartered Engineer and an active committee member of the BCS SPIN (UK) Special Interest Group. He is also a qualified ISO 9000 auditor and a qualified SPICE assessor.

Peter Henderson

Peter Henderson is Professor of Computer Science in the Department of Electronics and Computer Science at the University of Southampton in the UK. Prior to his move to Southampton in 1987 he was Professor of Computer Science at the University of Stirling, also in the UK. Henderson is also a visiting ICL Fellow. He is head of the Declarative Systems and Software Engineering Research Group (see http://www.dsse.ecs.soton.ac.uk/) which combines research interests in Software Engineering, Formal Methods and Programming Languages. His own research includes executable specifications, component-based systems, process modelling and the software development process. He has consulted for many companies, including ICL, on diverse topics in Software Engineering but in particular on Software Development Process Improvement. He has published two books and about thirty papers on Software Engineering. Currently Henderson is national co-ordinator for an EPSRC (Engineering and Physical Sciences Research Council) research programme entitled Systems Engineering for Business Process Change which has a legacy systems, COTS, component-based software development theme within it.

Chris Rostron

Chris Rostron is a Development Manager within ICL High Performance Systems. He is responsible for the planning; release management and quality assurance of a major UNIX based project. He has worked in the software industry for 30 years in various roles including programming, software design, release and configuration management, support including Product Introduction, business planning, and Quality assurance and project management. In the last 6 years he has focused on release and configuration management, planning and Quality Assurance. He gained qualifications in Cartography before joining the computer industry. He is a qualified ISO 9000 auditor and a qualified SPICE assessor.

Appendix 2: Company Descriptions

ICL is a leading supplier of IT systems and services. Operating in over 70 countries and employing over 19,000 people, the group's revenues for 1997 were £2,447 million generating a pre-tax profit of £30.0 million. The company implements IT systems for major projects and provides innovative services to a range of industries covering amongst others, retail, finance, travel, telecoms and utilities together with education and local and central government sectors. Its services include outsourcing, helpdesks, network services, inter/intranets, electronic commerce, interactive kiosks, smart card systems, digital cities and web sites. ICL plans to relist on the stock market in 2000.

ICL's website: http://www.icl.com

ICL's High Performance Systems Division (HPS) is responsible for the development, sales and marketing of enterprise-scale solutions and services. Working together with ICL business operations world-wide, our customer offerings are based on the Trimetra range, providing data centre solutions running OpenVME, Windows NT and UnixWare, and on i500, ICL's open directory software.

HPS is at the forefront of new technologies enabling the information society. These include interactive media servers, WWW-enabled software and advanced data centre solutions based on combining leading ICL and partner technologies and expertise.

Implementation of metrics in development of highly-safety critical SW

Kenneth Kvinnesland
Navia Aviation , Oslo Norway

Abstract

The Process Improvement Experiment AMPIC (Application of Metrics for Process Improvement for safety Critical software) has been carried out in the company Navia Aviation that consists of the former separate companies Normarc, Garex and Nova which have recently been merged. Navia Aviation is the largest exporter of Instrument Landing Systems (ILS) in the world. The product ranges also include systems for Air Traffic Control (ATC), Flight Inspection Systems (FIS), Coastal Radio Communication Systems, Radar Data Processing and Display Systems and Enhanced Surface Movement Radar Systems.

The fundamental objective in the PIE was to: Develop suitable metrics for SW-development in order to improve estimation, expose problem areas and also to improve the development process itself in respect of quality and productivity.

In the AMPIC-experiment software metrics has been derived from the overall company goals by using the Goal-Question-Metrics-Method The AMI approach which is very similar to GQM was used to assess the development process before the start of the PIE.

This paper described how the PIE was implemented in two very different organisations that used be separate companies. A brief description of the GQM method is given before the actual measurements and collected results in one of the baseline projects is described. The final section summaries the success and failure criteria in the two baseline projects

This section discuss why the PIE proved to be successful in one of the baseline projects but failed in the other.

1. Background.

Introduction

The AMPIC PIE was proposed by Garex and Normarc when they were still separate companies. It was however clear that the companies were going to be merged. Both experienced problems with completing the development projects within time and cost-limits. It was also evident that the quality of the development process had to be improved, as a new project involving development of highly safety critical software was about to be started.
This paper mainly focuses on the part of the PIE that has been successfully integrated with the Kappa-project at former Normarc. The part of the PIE that was integrated in a Coastal Radio-project at Garex more or less failed and the reason for this is discussed in the last section. The following sections gives an overview of the baseline project.

The Kappa-Project

The Kappa-project is developing the next generation of Instrument Landing Systems using satellite navigation based on the Global Positioning System (GPS). The product will in the future be a supplement to the existing ILS-product. Parts of the software developed in the project are highly safety-critical and shall be certified according to the standard DO-178B / ED 12B [1]. This standard defines different software levels based upon the contribution to potential failure conditions. Levels range from A to E with A being the most safety critical level. The certification is performed by the American Federal Aviation Administration (FAA) The safety critical software developed in the Kappa project will be certified according to level B which has the following definition:

Level B: Software whose anomalous behaviour, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a hazardous/severe-major failure condition for the aircraft.

The safety critical software is developed using a subset of ADA-83. The software is running on CPU-cards that are made within the company. No operating system is used.

Some issues that are expected to contribute significantly to the work load during development of this software are as follows:

- **Requirement traceability**: All software requirements must be traceable from their origin to source code. All code must be traceable backward to requirements or derived requirements which again must be traceable back to their origins. Derived requirements must be traceable back to specific design decisions.
- **Requirement based testing**: All requirements at all levels must be covered by test cases.
- **Coverage testing**: The test cases developed for the requirements must cover all source code and all conditions. It shall be proved that all source code has been exercised by the test cases in all conditions.

The amount of safety critical software is expected to be relatively small, between 5000 - 10000 lines of code.
Non safety critical software are developed using Visual C++ and will be running at a Windows-NT workstation. This software will be used for maintenance purposes. The amount of software is expected to be around 100 000 lines of code.

The overall goals for the PIE within the Kappa-project were to understand the impact related to development of highly safety critical software, particular in order to identify cost-drivers. Changes were expected to be very expensive, and special focus has been put on metrics related to change control in order to reduce the amount of changes to a minimum.

The Coastal Radio-Project

The project involves maintenance and further development of an existing software.
The basic part in the product is a digital telephone switch. The real-time software running in this switch constitutes 80 % of the total number of code-lines. The software is written in C. It runs on top of TST, which is a runtime system that serves as an abstraction layer between the operation system and the application.

New software projects are typically maintenance projects. I.e. the different customers require development of new functionality that are added to the existing software code base. The same code base is used for all customers, hence configuration management issues becomes very important.

The baseline project included development of the following new functionality:

- Software for Radio Control. I.e. remote tuning, diagnostics etc. for a specific set of radios.
- Software for a new switch card.

- Software for Pulse Code Modulation Diagnosis.

The overall goal for the PIE within the baseline project was to reduce the time and costs related to delivery of products, by improving specifications and by improving the configuration job done before delivery.

Recommended reading

The DO-178B/ED-12B "Software Considerations in Airborne Systems and Equipment Certification" is the bible for development of safety critical software in the avionics business. This document should be of interest for any company developing safety critical software.

References

[1] DO-178B/ED-12B, Software Considerations in Airborne Systems and Equipment Certification, December 1, 1992

2. The Goal-Question-Metrics method

Introduction

GQM is a method for breaking down overall goals into a set of factors that are measurable. The following sections contains an overall description of the GQM-method [2] as it was implemented in the PIE.

Initial phase

The initial phase involved the following steps:

- Identification of the overall goals.
- Performing interviews with the developers in order to expose problems related to the defined goals.
- Review of the material collected in order to define questions based on the suggestions, questions on hypothesis mentioned by the developers.
- Development of metrics from the material collected and the questions defined. This is graphically illustrated in a GQM-tree as shown in Fig. KKVAMPIC. 1 below.
- Validation of the results. The developers must validate the metrics in order to ensure that there is a common understanding about the problems, that the questions are relevant, that the selected metrics are relevant and that the measurements are possible.
- A measurement plan is developed based on the metrics.

These initial steps were carried out leaded by representatives from SINTEF Telematics which has long experiences with these kinds of processes. This assistance was crucial during the start of the PIE.

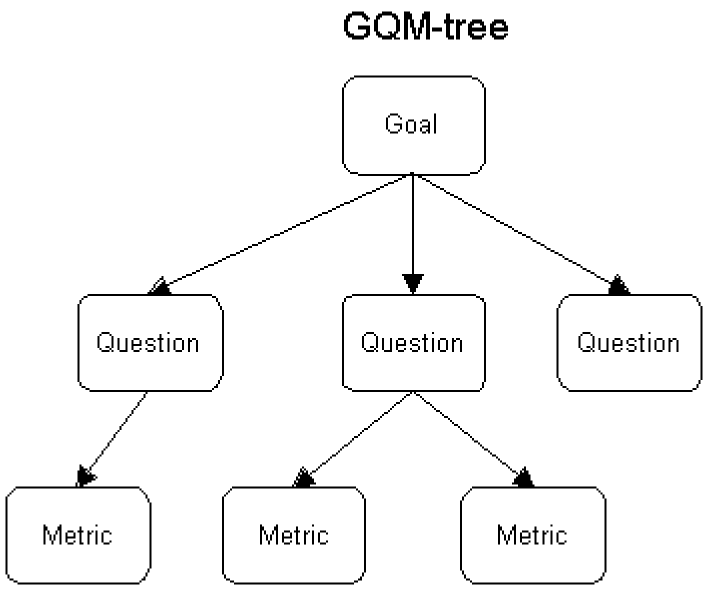


Fig. KKVAMPIC. 1: GQM-tree

Organising the information

Typically the information will be organised in a GQM Work sheet. An example is shown in Table 1. This sheet contains:

- A short goal definition.
- Quality Focus which contains questions that are developed from the goal definition.
- Variant Factors. This is a developed description of the environment presented in the goal definition.
- Baseline Hypothesis. This is what the developers think are the answers to the questions presented in Quality Focus before the actual measurements.
- Impact on Baseline Hypothesis.

GQM Work sheet			
Analyse:	Development Process	**In order to understand:**	Development of Safety Critical Software
Point of view:	R&D-department	**Environment:**	Kappa-project
Quality Focus		Variation factors	
Q2.1	What is the distribution of development costs among the activities ?	Q2.a	Type of sub-system • Safety critical • Not safety critical
Q2.2	What is the average cost of a change ?	Q2.b	Sub-system complexity
Q2.3	What is the volume of the source code ?		
Q2.4	What is the review results, in respect of the following subjects ? • No. of first time approvals • No. of pages • No. of remarks • Hours spent		
Baseline Hypothesis		Environment Impact on Baseline Hypothesis	
Q2.1	Type — Req. — Des. — Impl. — Test Not Safety Critical — 5% — 50 % — 30 % — 20 % Safety Critical — 30 % — 15 % — 5 % — 50 %	Q2.a.:	Safety Critical Sub-systems should: • Increase hours pr. change (Q2.2) • Increase percentage of hours spent in Requirements and Test (Q2.1) • Have a lower percentage of first-time approvals in review (Q2.4)
Q2.2	20 hours pr. change	Q2.b	Sub-system with high complexity should • Increase hours pr. change (Q2.2) • Have a lower percentage of first-time approvals in review (Q2.4)
Q2.3	5000 lines safety critical code 100 000 lines not safety critical code		
Q2.4	40 % approved in first reviews 20 Pages in average pr. document 20 Remarks (1 pr. page) 30-40 Hours (1.5 - 2 pr. page)		
Feedback			

Table 1 GQM Work Sheet

Definition of goals in the Kappa-project

The first major goal was to ensure that :

□1□The product quality is high enough to receive a certification by the American Federal Aviation Administration (FAA).

The product must be a "zero error product" and changes will be costly due to very strict change control mechanisms. Hence the following sub-goal was defined

G1.1 Reduce the amount of changes to a minimum.

The second major goal was to:

G2. Understand what it takes to develop safety critical software.

Development of safety critical software is expected to be more costly than development of regular software. In order to improve estimates and cost-control the following sub-goal was defined:

G2.1 Get an overview of the cost-profile related to development of safety critical software.

Recommended reading

The ami approach [3] is an elaboration of the GQM-method. It is very easy to read and should be beneficial for both developers and managers.

References

[2] Basili, Victor R., and Rombach, H. Dieter, "The TAME Project: Towards Improvement-Oriented Software Environments", Institute for Advanced Computer Studies, University of Maryland, UMIACS-TR-88-8, January, 1988.

[3] Application of Metrics in Industry a quantitative approach to software management South Bank Univ., London 1992 (ISBN NO. 0 9522262 0 0)

3. Measurement in the Kappa-project.

Introduction

This chapter describes the methods used to collect the data and it shows a representative selection of the results found in the Kappa-project so far. It also describes some of the process improvements that has been implemented.

Methods for Data Collection

Existing forms of reporting was improved in order to make reporting overhead as small as possible.

- A new database based on Microsoft Access was developed. This database is used both for definition of work packages and for registration of time spent in the different activities. Each work-package is divided into classes of sub-activities, E.g. requirements, design, test, review , rework etc. This makes it easy to make queries on these classes in the database.
- A new problem report database based on Sybase SQL has been customised in order to support collection of metrics related to the change process.
- Review reports have been changed in order to collect metrics related to the review process.

Analysis of data from the Change Process

These data are used to answer the following questions:

- What is the distribution of changes among the different type of changes?
- What is the distribution of changes among the project phases where problems were introduced ?
- What is the distribution of changes among the causes of the changes?
- What is the distribution of changes among the project phases where problems where found?
- What is the distribution of changes among components in the product structure?

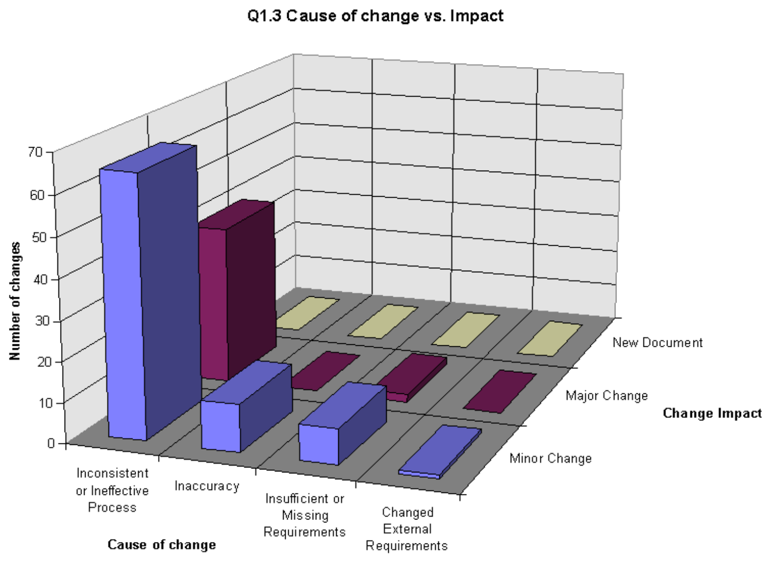


Fig. KKVAMPIC. 2 : Cause of change vs. Impact

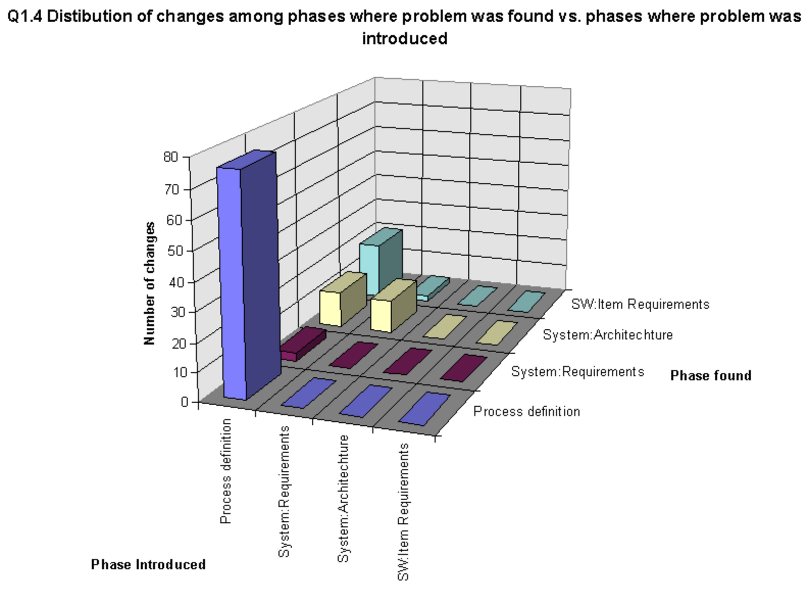


Fig. KKVAMPIC. 3 : Distribution of changes among project phases

The examples shown in Fig. KKVAMPIC. 2 and Fig. KKVAMPIC. 3 shows parts of the current status regarding the questions listed above. Most of the changes implemented so far has been related to the documents defining the SW-process. Only a small number of changes have been implemented in the system requirements and system architecture. The data from the change control activity will become more interesting as the project moves into the coding phase and the test phases.

Analysis of data related to costs.

These data are used to answer the following questions:

- What is the distribution of development costs among the different types of activities ?
- What is the average cost of a change ?

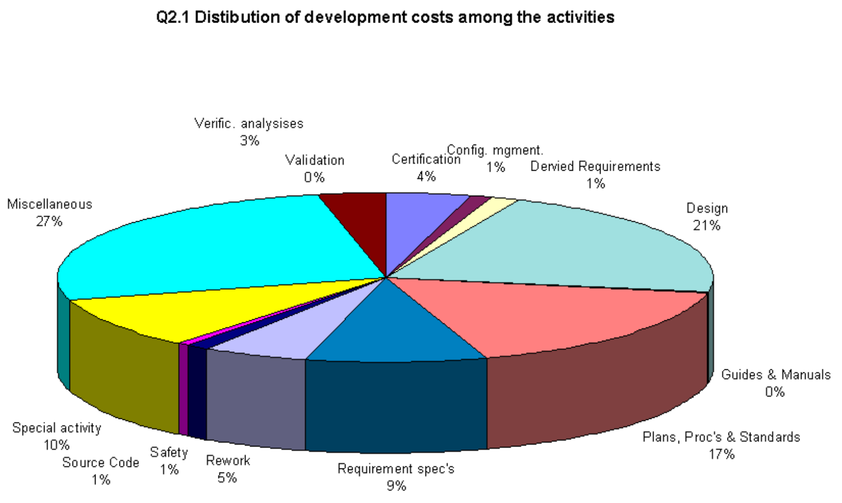


Fig. KKVAMPIC. 4 : Distribution of development costs

The distribution of different classes of sub-activities in the Kappa-Project is depicted in Fig. KKVAMPIC. 4.

- The total amount of time spent in the project is now about 12000 hours.
- The classes Miscellaneous and Special Activity contains among other things project management, training, and development of tools. These data are now being re-classified in order to make the real content more visible without expanding the level of details.
- The class "Safety" refers to the Functional Hazard Analysis (FHA) which is being performed by external consultants, which explains the low percentage.
- 95 % of the rework so far is related to plans and standards.
- The total effort related to SW process development and certification is so far about 35 % or 4200 hours. This also includes development of a requirement database. It is now expected that the total effort will be about 10 000 hours.
- It is still to early to distinguish between the safety critical modules and the non safety critical modules.

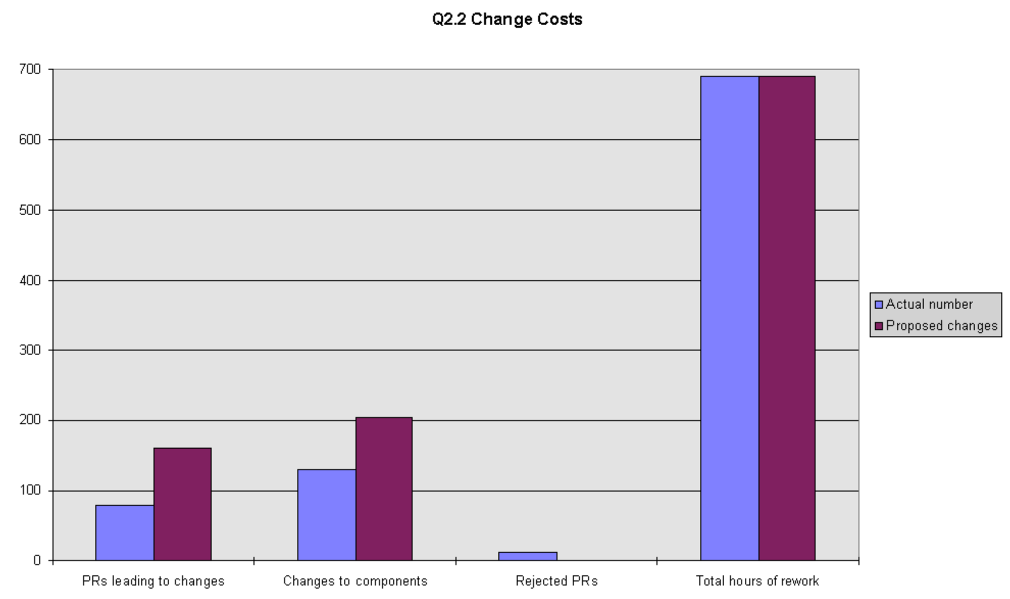


Fig. KKVAMPIC. 5 : Change Costs

Fig. KKVAMPIC. 5 shows the number of problem reports that have been resolved and how many changes that was made in this process.. The cost for each problem report is so far 8.6 hours. These data confirms that the strict change control process which is necessary to comply with [1] is very expensive. The project continuously tries to make the process more smooth and to make the review process more effective as described in the next sections.

Evaluation of the Change Process

Evaluation of the costs related to the change process revealed the following problems:

- There were two boards involved in the process, the Change Control Board (CCB) and the Release Control Board (RCB). This organisational model was based on experiences from another company developing safety critical software. The CCB handled the administration of the process while the RCB became involved if a problem had major consequences or was related to management issues. This model did however create a significantly amount of overhead.
- The quality of the change orders were not good enough. Too many was rejected by the CCB and too many proved to be impossible to implement.
- There was too much overhead in the communication between the CCB and the engineers which investigates and implements the changes. Problem reports and change orders were sent back and forth between the board and the engineers too often.

Improvements in the Change Process

Based on the evaluation the following improvements were made:

- The RCB was merged with the CCB. The new CCB now a number of permanent members while other members attends the board meetings as required based on a set of business rules.
- Change orders are now pre-reviewed before being assessed by the CCB in order to prevent rejection at the board meeting.
- The investigation process and the change process are now merged when possible to reduce the overhead in communication. In the old model the majority of problem reports were assigned to engineers for investigation. The assigned engineer returned a proposed solution to the CCB which would decide whether to include it in a change order or not. In most cases however the CCB did know in advance if it was necessary to issue a change order on a specific configuration item. In the new model the CCB may assign an expanded action to an engineer. Typically such an action includes responsibility for the investigation of the problem, preparation of a change order and implementation of the changes. Consequently a problem report is not returned to the CCB before the change order is implemented and closed. The problem report database has been upgraded to match these changes.

A new change control standard which incorporates these changes was approved in September 1998.

The Document Approval Process

There are 6 different roles defined in the approval process, these are: Author, Inspectors, Review Moderator, Subject Responsible, Configuration Management (CM) and Software Quality Assurance (SQA). The Review Moderator approves the review process and the Subject Responsible approves the technical content of the document. Finally the document is approved by CM which checks that all material have been archived according to the CM-plan, and by Software Quality Assurance (SQA) which approves the process as a whole.

The review process is based on the principle of formal inspection, but has some special characteristics:

- The Review Moderator is always a person that is up to date on the technical content and is very often identical to the subject responsible.
- The author does not have any formal responsibilities for the content of the document.
- The review meeting is democratic, I.e. all the inspectors may put a veto on approval.
- Discussion at the review meeting is allowed.
- Most of the issues are characterised as Minor. A Major issue automatically leads to a second review.

Analysis of data collected in review.

These data are used to answer the following questions :

- What is the approval rate ?
- What is the average no of pages ?
- What is the average no of discrepancies pr. page ?
- What are the costs of a review including preparation?

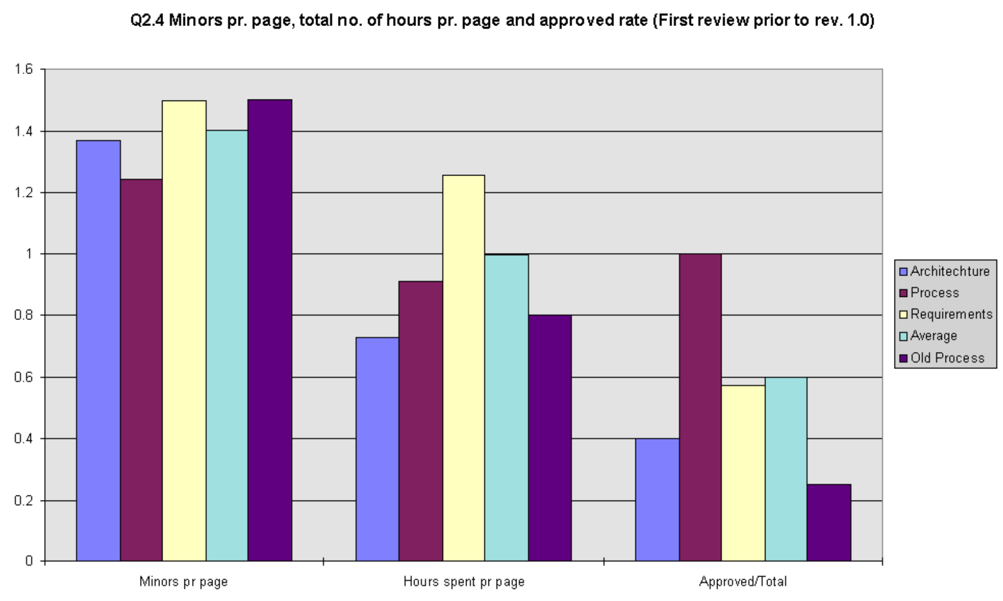


Fig. KKVAMPIC. 6 : Errors, costs and approval rates

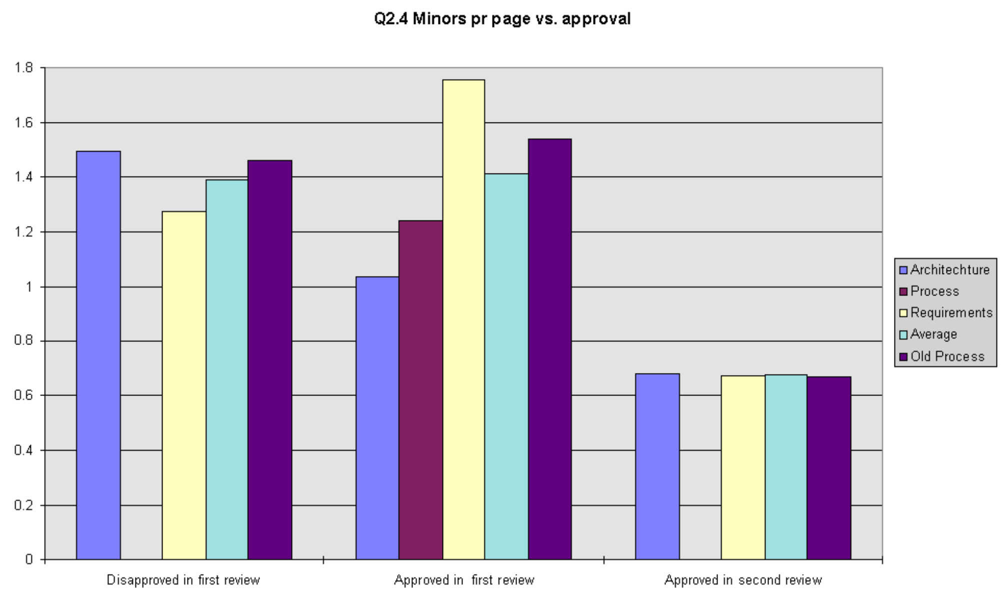


Fig. KKVAMPIC. 7 : Results from reviews vs. results from re-reviews.

The examples shown in Fig. KKVAMPIC. 6 and Fig. KKVAMPIC. 7 shows examples of collected data related to the approval process. The term Old Process is used to identify the first series of review which was performed on the plans and standards. The review process was then changed because only 25 % of the documents where approved in the first review as described in the following section.

Evaluation of the Approval Process.

Evaluation after the first analysis of the data collected revealed that the inspectors was not strict enough when they participated in walk-through prior to formal inspection. This meant that many immature documents were subjected to inspection and consequently the approval rate was very low (25%). The duration of the review meetings was too long and there was also too much discussions. This reduced the quality of the review meetings. Discrepancies were unnecessary often defined as Major Issues which again led to a second review. This happened because the inspectors wanted the solution of all discrepancies to be 100 % specified in the review meeting.

Improvements made to the Approval Process

Based on the evaluation the following improvements were implemented:

- Pre-review: A Competent person appointed by the Review Moderator must review the document and give go for formal inspection.
- Discussion is only allowed at the end of the review meeting, I.e. the list of issues shall be fully recorded before discussion about the solutions is allowed
- A review meeting should not last more than two hours. Statistically a review meeting manage to do 8 pages pr. hour on a new document. Consequently a reviews of large documents must be split into several meetings.
- An issue may be defined as "Minor", even if the solution of the discrepancy cannot be 100 % defined in the review meeting. In this case the solution must be approved by the Moderator or the Subject Responsible.

After these changes a new set of data was collected. The main discovery was that the approval rate in first review increased from 25 % to 60 % and this showed that the quality of the work done prior to the review had improved.

However the review efficiency was not increased, I.e. the percentage of discrepancies found during first -time inspection had not been increased and is still only 50 %. The project is working to improve this number.

The results also shows that the number of discrepancies found in review is remarkably stable regardless document type, especially in re-reviews. Review of requirements is more expensive than other review and this result was expected due to the very strict verification objectives for

requirements. Some of the requirements documents approved have a relatively high number of discrepancies pr. page compared to the average, evaluation shows that this was due to high time-pressure. It is expected that these documents will be subject to more changes than the average documents because they are more immature.
Based on these results the project tries to increase the time the inspectors spend in preparation and also to put more emphasise on walk-troughs prior to the inspections. The project is also considering objective criteria for approval based on statistical data, I.e. to put a specific limit on number of discrepancies pr. page.

Summary of experiences regarding safety critical software.

The main experience is that the effort of developing a development process that complies with [1] showed to be much more difficult than assumed.

- Currently more than 4000 hours has been spent on development of plans and standard and the certification process in general. This effort was substantially underestimated at the start of the project. However this also means that the company has invested heavily in knowledge about SW development processes.
- The data collected so far confirms that the strict change control necessary to comply with [1]makes changes very expensive. Some improvements to the change process have already been made, but the data from the change control activity will become more interesting as the project moves into the coding phase and test.
- By analysing the statistical data and do some simple changes to the review process it has been possible to improve the process. The average number of approval on first attempt has increased from 25 % to 60 %. The process should however still be improved because analysis of data shows that only 50 % of the discrepancies are found in the first review. This process is the key to reducing the change costs.

Recommended Reading

"Developing a Successful Metrics Program" [4] . This paper demonstrates how GQM can be used in a small metrics project. The method is applied (in theory) on data collected at the NASA Goddard Space Flight Centre (GSFC).

References

[1] DO-178B/ED-12B, Software Considerations in Airborne Systems and Equipment Certification, December 1, 1992

[4] Rosenberg, L and Hyatt L. "Developing a Successful Metrics Program" resented at 8th Annual Software Technology Conference, Utah April 1996

4. General experiences

The following sections discusses the differences between the baseline projects and tries to explain why the PIE became a success in one project and a failure in the other.

Maturity of the existing processes

The GQM-method was applied in the same way in both the two baseline projects.
The situation in the two projects was however very different. The investigation in the Coastal Radio-project revealed that the existing development process was much more immature than originally assumed. Many of the leaders at a lower level in the organisation felt that the process and the organisational environment was too unstable for measurement. There was also an immense time pressure in the organisation because several projects was delayed. The R&D manager who did initiate the PIE was a driving force but otherwise motivation was low.

The environment in the Kappa-project was much more favourable, even if there were a great number of risk factors involving both new technology and a new development process, The Kappa-project had just started to define a development process that was going to comply with [1] when the PIE started. In fact there was no existing development process at the starting time. Many of the developers had however been working with processes improvement within other departments in the former Normarc.
At the start of the PIE there was not any great time-pressure in the project. The work with the SW development process had absolute priority because of the certification. in addition the project manager for AMPIC was also responsible for the development of the process in the Kappa-project. Consequently metrics became an integrated part of the development process from the start. The project manager for the Kappa-project at that time was also very focused on the development process.

Cultural differences

There are major cultural differences between the two projects which explains why the people working in one of the baseline project was reluctant while the people in the other department did welcome the changes:
During the last years there has been several attempts to improve the development process in the organisation running the Coastal

Radio-project. The major problem is that it has always been the management that has tried to make changes and that low priority has been given to such issues compared to other projects in the R&D-department. The developers tend to be tired of "bright ideas from the management", and it has been difficult to convince them about the value of the PIE.

The R&D-section running the Kappa-project has a another history. Many of the management issues related to software have traditionally been handled by the developers themselves. Historically this is because the number of people working with software used to be relatively low, and because few other people in the organisation had any knowledge about software development. Because of this, the software developers had a high degree of influence on management, and they have been a driving force behind improvement in the software development process in other departments also prior to the PIE.

Unstable organisational environment

There were great problems due to turmoil in the organisation running the Coastal Radio-project. In many ways the merging of the companies became a take-over and this created a cultural shock which at times lead to a quite hostile climate between the management and the employees. During the process both the former General Manager the R&D manager which originally proposed the PIE did resign. This took away much of the driving force from the PIE. The new management did support the PIE but it did also initiate a lot of other change processes and the baseline project was redefined and rescheduled.

Timing problems between the PIE and the baseline projects :

Timing became the worst problem for the PIE because of major delays in both the baseline projects. For the Coastal Radio-project this was mainly due to other development projects that did not finish in time. It took a long time after the metrics were defined before the baseline project actually started and it became difficult to keep the subject hot.
The Kappa project had a much longer "warm up phase" than planned. Full effort was not started until August 97 and the first 6 months was completely dominated by the effort of defining a development process compliant with [1]. The PIE was however very successfully matched with this work, and the advantage was that measurement was integrated in the new process from day one. However the Kappa-project itself did not start producing data relevant for the PIE until January 98. Consequently it was difficult to give feedback to the developers and motivation problems occurred also in this project. Another effect is that much of the most interesting data will be produced after the PIE has finished. Results will however be made public available also in the future.

Report policy

The nature of the metrics that was defined in the Coastal Radio-project became a problem. Many of the metrics was supposed to be collected when the developers experienced difficulties. This also meant that the developers could decide whether to report or not.

In the Kappa-project on the other hand the developers have to report because reporting is an integrated part of the tools used to register time spending, problem reports and review results . The quality of these reports may be more or less accurate dependant of the individual motivation, but it is impossible not to report. The use of the database tools has also made reporting overhead very small.

Summary of the success factors

- The development process must be stable before metrics is applied..
- The organisational environment should also be relatively stable. Turn over and re-organisation makes it very difficult to keep a stable development process.
- The PIE should be adapted to the pace of the baseline project. It is very difficult to synchronise the 18 months duration of the PIE with the schedule of large development projects which tends to be delayed. Process improvement should preferably be a background activity which makes a small but steady progress.
- There must be a least one impelling force working in the baseline project. This person should preferably be a key figure among the developers. It is very difficult to make any progress without such persons.
- The assistance from SINTEF Telematics was absolutely necessary in order to get started with the project.
- Regular feedback to the developers is necessary to secure accurate reporting.
- Report overhead must be reduced to a minimum. Direct reporting in SQL-databases has proven to be successful.

The Kappa project is still in an early phase, and the most interesting experience data produced in this project is yet to come. Navia Aviation will continue to spread experience data from this project also after the AMPIC PIE is completed.

Recommended reading

"Implementing Effective Software Metrics Programs" [5]. This paper analyses success and failure criteria found in two organisations that was both running metrics projects.

References

[5] Hall T. and Fenton N. "Implementing Effective Software Metrics Programs" IEEE Software March/April 1997.

Session 5 – Implementation of SPI Part II

Lessons learned in a National SPI Effort

The Danish SPI initiative:
Centre for Software Process Improvement

Jørn Johansen
DELTA Software Engineering, Denmark
Lars Mathiassen
Aalborg University, Denmark

SPI by IPS - Involvement, Planning, Structure

Bill Culleton
Silicon and Software Systems (S3), Ireland

Experiences from practical software process improvement

Seija Komi-Sirviö, Markku Oivo, Veikko Seppänen
VTT Electronics, P.O. BOX 1100, FIN-90571 OULU, FINLAND
Seija.Komi-Sirvio@vtt.fi, Markku.Oivo@vtt.fi, Veikko.Seppanen@vtt.fi
fax: +358 8 551 2320, tel. +358 8 551 2111

Lessons learned in a National SPI Effort

The Danish SPI initiative:
Centre for Software Process Improvement

Jørn Johansen
DELTA Software Engineering, Denmark

Lars Mathiassen
Aalborg University, Denmark

Abstract:

This paper focuses on experiences from the first part of a Danish, national research- and collaboration initiative on software process improvement (SPI) involving four software organisations and a dozen researchers. The general experience is that Danish organisations can benefit from SPI initiatives. Such efforts are, however, resource demanding, they require a high level of management commitment and participation, and they typically involve fundamental changes in the software processes and environments. This paper presents a number of practical lessons focusing on learning to practice SPI, on taking advantage of the key features related to SPI, and on dealing effectively with the organisational changes involved in SPI initiatives. Other parts of the experiences are published as separate, scientific papers.

A National SPI effort

Software development is a young discipline. Although our knowledge has grown in the last 30 years there still is a pronounced need for improvement in quality and productivity in software development. Continuous improvement efforts have become a strategic necessity to maintain profitability and meet the requirements from the market. Today the most forceful and promising approach for improving the software development process is based on maturity models such as CMM [16] & [27] and BOOTSTRAP [22].

A national initiative has been formed in Denmark to facilitate the use of such approaches within the software industry. This collaboration runs over three years (1997-1999) with a 2.6 Million ECU budget, of which half is financed by the Danish government: Ministry of Commerce (Council for Development of Business and Industry) and Ministry of Research (Centre for IT-research). The initiative includes four companies, Brüel & Kjær A/S, Danske Data A/S, L. M. Ericsson Denmark A/S and Systematic Software Engineering A/S, and also DELTA Danish Electronics, Light & Acoustics, Aalborg University, and Technical University of Denmark. The missions of the initiative are:

- To systematise SPI knowledge in Danish companies.
- To tailor and further develop the most promising models for SPI, so they apply for the Danish software industry.
- To develop frameworks for managing, organising and implementing SPI activities in Danish companies.
- To communicate and publish knowledge about SPI to Danish companies.

These missions are addressed through action research efforts in each of the four software organisations in which the following basic questions connected to software process improvement are addressed:

- **Modelling:** How can we understand software development processes, the conditions under which they are performed, and their capability to develop quality software?
- **Measurement:** How can we assess the capability to develop software, identify the appropriate improvement areas and strategies, and measure the effect of the implemented improvements?
- **Change:** How can we manage, organise and carry out concrete and sustainable initiatives to improve an organisations capability to develop quality software?

The research and development collaboration has been organised around:

- Four research groups at the participating companies.
- A common research forum across the companies.
- A Danish network on software process improvement.

At each of the four companies, a local research group is established. This group works tight together with the management, the local SPI-group, and the ad-hoc groups established to work with specific improvements, cf. *figure LM-JoJ.1*. A research group includes 4-7 employees from the company (normally the SPI-group), who are involved in or has responsibility for improvement activities in the company, and 3-4 external researchers. The research group meets 8-10 times a year, and the initiated improvement work in the concerned company is followed closely:

- It supports the company in adapting and using improvement methods.
- It participates in the companies' preparation and implementation of improvement activities on identified improvement areas.
- It assesses strong and weak aspects of methods and the way in which they are used.

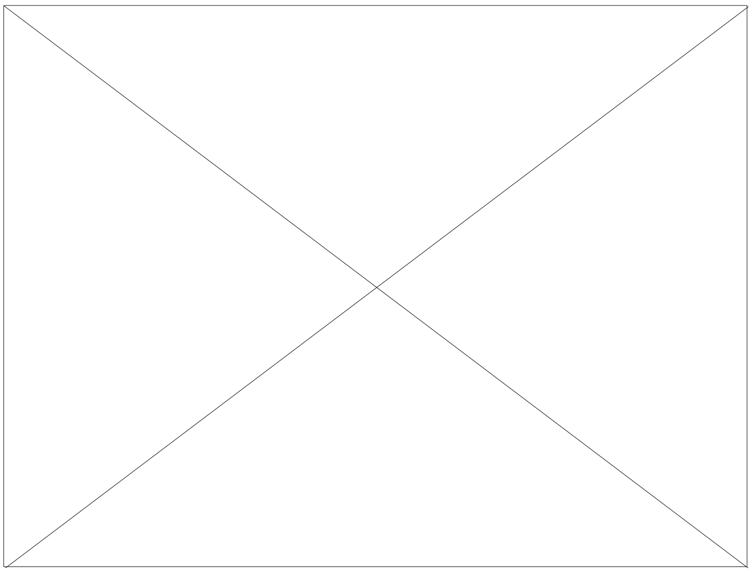


Figure LM-JoJ.1: Organisation.

The research forum consists of the four research groups. This forum meets twice a year in common workshops where:

- Experience and knowledge is exchanged across the companies.
- New process improvement knowledge is presented.
- Inspiration for specific improvement activities is presented.
- Improvement activities are put into perspective through general discussions about software management and organisation development.

In addition, the project manages and supports an open network for Danish software development companies who are involved in SPI activities.

Lessons learned

In the following, we present ten lessons based on the work performed in the four companies and documented in a midterm report for the project [40]. Experiences have as a first step been systematically collected, discussed and documented in each of the participating organisations. These experiences are derived from important episodes, encountered problems, and successful initiatives documented as historical data in each software organisations. Subsequently, these experiences have been compared and contrasted across the four organisations to arrive at more general knowledge on SPI. The resulting lessons focus on learning to practice SPI, on taking advantage of the key features related to SPI, and on dealing effectively with the organisational changes involved in SPI initiatives.

Data-driven interventions motivate and provide insight

Process improvement initiatives are based on measurements, which give knowledge about the maturity of a development process in a company or indicates the effect of the process improvement initiatives. The improvement initiatives are therefor not primarily driven by new ideas or technologies. They start from systematic knowledge about existing practice and about the effect of earlier initiatives. For most companies this requires a change in both attitude and management practice in relation to methods and new technology.

Maturity assessments have played a decisive role in the early progress of this project. Insight in strengths and weaknesses has formed the basis for planning the improvement activities and has strongly motivated both management and developers in the work with improvement of the software development process.

A specially developed maturity assessment approach, called Problem Diagnosis, was used at Brüel & Kjær, especially to involve the project leaders knowledge and experience. This strategy has been vital in establishing a solid basis and a positive climate for improvement.   At L. M. Ericsson the use of self-assessments, performed by the project leader or the project group, has given valuable insights into existing practices and at the same time motivated projects to engage themselves actively in improvement activities.

So far there are only few experiences with measurement on effects of improvements. The main reasons are that it takes time and requires many resources to establish a well functioning effect measurement programme. The first experiences with presentation of effect measurement results are, however, positive. At Danske Data, effect measurements were implemented in tight co-operation with management to achieve a common understanding of the existing processes and to establish a fruitful and well functioning measurement programme. The first data caused interest and gave rise to debate with top management. The quality of the data provided are, however, still insufficient to be used as practical tools in the improvement activities.

A discussions of measurement during software process improvement work are to be found in [13], [16], [28], [42], [43] and [44].

Maturity can be assessed in different ways

Maturity assessments have been practised in all four companies using several different assessment methods: CMM and BOOTSTRAP as the main tools, but also QBA (a level 2 CMM Questionnaire Based Assessment method developed by Aalborg University [43]), CMM Light Assessments, Ultra Light Assessments and Problem Diagnosis. *Table 1* shows which methods have been used in combination. The assessments have established that all four companies should either work on reaching level 2 or consolidate it.

Table 1: Used assessment methods in the four companies.

Company/Method	**Brüel & Kjær**	**L. M. Ericsson**	**Danske Data**	**Systematic**
CMM		⚐		
BOOTSTRAP	⚐		⚐	⚐
QBA			⚐	⚐
CMM Light		⚐		
Ultra Light		⚐		
Problem Diagnosis	⚐			

The different assessment methods have only lead to minor differences in the derived recommendations, but they each have their strengths and weaknesses. CMM and BOOTSTRAP are comprehensive methods supporting an intensive and quite visible assessment process in the organisation. QBA, CMM Light and Ultra Light Assessments are less intensive and can therefor be used more often and broader in the organisation. These methods are less accurate due to their reduced extent. Problem Diagnosis is directed towards a well-defined group in the organisation and provides a picture of their understanding of the situation. To combine these approaches it is important to consider the main purpose and the participants in an assessment.

Comparing the use of questionnaires with interviews it has been unambiguously reported, that the interview form gives more engagement and a higher level of quality of the answers.

More discussion on this subject is to be found in the following litterature: **CMM** [27], [2], [6], [5], [9], [23], **BOOTSTRAP** [22], [3], [14], [18], [45], **SPICE** [29] and the **Problem Diagnosis** [44].

Establishment of a metrics program is a project in itself

To reach a higher level of maturity can be an objective in itself for some companies. However, most organizations want to know whether the improvement activities contribute to specific objectives for the company or its employees. In these organizations, there has, for example, been a pronounced wish to know if there is a change in customer satisfaction, error rates in products, productivity, or motivation among employees.

None of the four companies have so far established a well-functioning effect measurement programme although some of the companies have spend a great deal of resources to do so.

One encountered challenge relates to creating inexpensive routines for data collection which at the same time give satisfactory coverage and validity. This task has in all the companies appeared to be much more complex than expected. It is therefor strongly recommended to define effect measurement programmes as an improvement project with its own plans, resources, and obligation to document improvement. The initiatives will be more visible and they can be organized in an iterative fashion in which improvements of the measurement programme become essential parts of the overall improvement project.

Diffusion of the effect measurement results represents a separate challenge. On one side it is inexpedient to publish measurement data, which are not sufficiently reliable. Many will question their value and the discussion will therefor focus on critique of the measurement programme instead of improving the understanding of present practices in the company. A publication of the measurement results can, on the other side, contribute to a higher validity, partly because the projects then become more serious providers of data, and partly because publication can initiate discussions which identity weaknesses and point at feasible and practical solutions. In general it is recommended to practice early publication and diffusion of measurement data.

Other experiences on effect measurements and Return on Investments can be found in: [4], [10], [19], [47], [21] and [34]. [27] presents a guide in a goal driven measurement programme for software development.

Process improvement is organisational development

Englebrecht suggests that organisational development efforts include five distinct efforts (Engelbrecht, et al., 1991) First, the vision and motivation behind the effort is communicated to the organisation and an initiative group is established. Second, a situation analysis is performed in which a shared picture of the future is created. Third, this picture is expressed as a number of prioritised objectives based on broad participation to establish the necessary motivation in the organisation. Forth, these objectives are turned into specific improvement activities involving experiments and operational planning. Finally, the initiatives have to be implemented as parts of a new organisational practice. In this project, most of the effort in the companies have been on efforts two to four above:

- Analysis of the actual situation (assessments).
- Specific improvement activities and experiments.

This is not to say that nothing has happened concerning the other aspects of organisational development, but they do not stand out as visible as these do until now.

The other three types of efforts involved in organisational development have been addressed—visions and strategies related to improvement of the software development process have been developed and presented, initiatives to make others participate in prioritising initiatives have been taken (and always with success), and some efforts to implement improvement initiatives have been completed—but, following the dominant rhetoric of the SPI literature, each of these efforts have so far played a less dominant role than the two mentioned above. In their continued efforts to create successful and sustainable improvements each the organisations are now focusing broader on all the concerns involved in organisational development.

Inspiration can be found in e.g. [1], [36], [38], [20], [51], [25], [26] and [31].

Process improvement initiatives should set an example

It is important that a process improvement project appears as an example of efficiency and competent work. An often-used slogan goes like this: ”A project for process improvement must practice at least one level of maturity higher than the organisation which is undergoing an improvement process”.

Our measurements show that the companies’ maturity level is 1 or very close to level 2 on the CMM scale. The improvement projects should therefor act as being on maturity level 2 and show professional project management. In practice this has proved difficult. The projects have suffered from frequent replacements of key employees, when they were missing in development projects or for other top priority tasks. Problems with setting precise goals and supply of necessary resources have also been very typical for process improvement projects.

At Danske Data, the improvement project is characterised by concrete goals at the organisational level, powerful management back up, a well-qualified crew and a few dedicated resources together with people involved on a part-time basis. In spite of this, the project has been suffering from unrealistic planning, problems with focusing the SPI efforts, and difficulties in getting the right key people involved in implementing initiatives.

At L. M. Ericsson, many synchronous improvement activities have taken place resulting in insufficient attention on each single initiative and hence too little progress. Yet, in a period the focus was on preparing for a formal, company-wide CMM assessment to qualify for level 2, and in that period visible progress was made.

It has been quite difficult for all the involved organisations to make SPI initiatives function as an example and there is a good reason to believe that this problem will arise in most process improvement initiatives. Especially companies at the low maturity levels can find inspiration in [13] and [24]. Other related experiences can be found in [7], [8] and [15].

Process improvement must lead to visible results

Process improvement takes time. It is, therefor, important to demonstrate substantial results during the project. Only by visible improvement, results, and progress, is the improvement project able to demonstrate and remind the organisation about its worth. This is especially important in relation to the motivation among management and employees.

Creation of visible results has appeared to be an essential problem in all the companies. Both management and developers have been curious to see and experience concrete results. An important reason for this is, that expectations and plans have been quite optimistic and ambitious (unrealistic), and the objectives were not divided into manageable subobjectives, which could demonstrate progress.

Although it is difficult, several visible results has been achieved so far:

- At Danske Data, an extensive effect measurement programme is under implementation, a competence centre has been established for project managers, and a large diffusion and adoption project has influenced the organisation and management of diffusion activities in general.
- At L. M. Ericsson, intensive use of Ultra Light Assessments have resulted in increased motivation and participation which has led to improvements corresponding to one step on the CMM scale.
- At Brüel & Kjær, new development models are under implementation, new project follow-up procedures and tools are now in use, and new processes for requirement specification are experimented with in a number of projects.
- Systematic Software Engineering has developed new process descriptions for project and configuration management, and updated descriptions for review and risk management. Adoption of these procedures is supported by an intensive project management course involving all project managers.

Other experiences on this subject are discussed in [13], [24] and [34].

Process improvement requires dynamic organisation

All four companies recommend that improvement projects should be implemented as normal development projects. This means planning, including a project plan with time schedule, milestones, activities, and intensive follow-up in relation to a requirement specification and the project plan. The reason why the companies emphasise this is because planning and control in reality is a problem.

The four companies all have symptoms of problematic planning and control. Planning has been at a too high level or even missing which have resulted in insufficient resources, weak co-ordination, and insufficient communication about the activities in the improvement project with the rest of the organisation. A missing plan for a project signals less importance and low priority in a busy everyday life where improvement activities are not given the same (or higher) priority as development activities.

All things considered, the recommendation from the companies is as clear as it is correct: The involved parties must be able to influence the project plan, they must commit to it, and the required conditions must be in place. During the project, it has to be clear what the project delivers and it has to be simple to verify whether the defined goals are met in the end.

In each of the four organisations we have experienced projects as the ideal form to organise improvement activities. The existing departments for new methods and tools have been questioned and this has led to several reorganisations of the activities related to technological innovation and support. In Systematic Software Engineering, the existing Quality Assurance Group had to be merged with the SPI initiatives resulting in a number of reappointments and reorganisations. In Danske Data, the centralised method and technology department was reorganised into a number of local groups and a smaller centralised group, and all the innovation activities where co-ordinated as part of one comprehensive SPI initiative. In general, the organisation and planning has to be as dynamic as possible to be able to accommodate changes needed as result of the ongoing learning involved in become engaged in SPI activities.

[13] and [24] give usable advises on how to manage a process improvement project. [11] and [30] focus on organisation of and responsibility for SPI-groups.

Management must play an active role

Commitment from management is often mentioned as an absolute necessity to obtain success in SPI. Commitment from management has three dimensions:

- *Insight* in what SPI comprises and what it means for the organisation. This includes insight in the process improvement project, e.g. through involvement in follow-up on the project.
- *Support* in the form of internal marketing of SPI, ensuring of stable resources, definition of requirements to the improvement project and to the persons whom are going to use the results. It also includes active involvement to find solutions in conflicts of interests.
- *Accept* of the improvement project through involvement in preparation of the project plan, including a time schedule, budget, and allocation of qualified resources.

Commitment is not just to start an improvement project and give the project team responsibility for improvement. Commitment includes build-up of sufficient knowledge about SPI among managers, to be able to match the project team in defining the vision, the expectations and requirements, and also to give the project a competent follow-up.

In several of the companies an improvement area was defined, an activity was specified, but time and resources were subsequently not allocated and too seldom were specific improvements required by management in the product development projects. A continuous pressure from management contributes to support a positive attitude for change. The priorities performed by management are the priorities, which the employees tend to comply with. If product development is prioritised higher then process improvement, then process improvement work will be of rudimentary importance. During the first part of this project the companies have realised, that success with software process improvement is tightly coupled to management attention and follow-up, which has to be equal to product development projects.

This subject is also discussed in [13], [24] and [17].

There are many roads to improved project management

Improvement of project management is a central issue in software process improvement. Partly because good project management is a condition for improvement of other processes, and partly because many problems in software development are connected to project management. Experiences across the companies are very different, and the overall experience is, that there are many roads to improve project management.

Systematic Software Engineering started their improvement of the Project Management process by establishing a group consisting of department managers and experienced project managers, who should formulate procedures for project management. This preliminary work has now been ongoing for 1 year. The procedures are finished, but not yet fully implemented in the organisation. Implementation is closely coupled to an ongoing education of all project managers in the company.

Danske Data has chosen to establish a competence centre for project managers, in which the project managers have the responsibility for their own competence development. The experiences with written procedures are not especially good in the organisation. Because of this involvement has been in focus under construction of this centre. Firstly, the competence centre is based on the project manager’s own responsibility for contents – it is their problems and their involvement, which are the substances in the centre. Secondly, a list of 10 objectives for project management is formulated, which include a standard for good project management practice. Thirdly, the centre offers education to build up competence. Status is that the competence centre is fully established, an incentive scheme is agreed by the management, and the education plan is in place.

Brüel & Kjær have for several years worked for improvement of project management, primarily the “soft” parts of the project management process, such as teambuilding. The key to change at the company is to solve the project manager’s problems. Especially two areas have been in focus: Project follow-up and a more iterative development model. An experiment with ‘Time-boxing’ is started, with the purpose to show if it is an effective control technique for the company. In connection with project follow-up, an early decision has caused purchase and installation of a new and customised project management tool. Success is conditioned by the project manager’s experience with this tool. They have to feel they get help, not bureaucracy - an experience from the PRIDE project [33].

At L. M. Ericsson the project management improvement initiatives are controlled by the requirements from CMM level 2. The project managers have to fulfil these requirements. Each project manager is responsible for the management as well as for the improvement of the project management process in the project.

Hence there are many, very different ways in which project management can be improved. Which one is the most practicable way depends on which requirements, traditions, successes, and failures the company has.

Essentially and related factors are discussed in [12], [15], [13] and [24].

Each organisation must make its own experiences

The improvement projects have been followed intensively by persons with both theoretical and practical knowledge in software development and process improvement. There has been no lack of advice and recommendations from these persons. At the companies, there are persons with concrete insight in process improvement inside the company. Nevertheless, experiences and not least mistakes seem to play a major role in an improvement project. “We were possibly able to read about it in a book, and somebody has very likely said it, but nevertheless we fell through at this point”, a manager said at a workshop.

There are several reasons for this. To get people to do something is a challenging communication process, in which various ingredients, such as trustworthiness, arguments, and feelings are included: “We are special” or “We know best what is the best for this company”. Besides, basic communication problems, political games, and power struggles each play their important roles in any SPI initiative. Minefields in the shape of de-motivation therefor surround the improvement projects.

No panacea is available, but time used to discuss and exchange experiences with the projects is well used. With the specifics of the situation at a company in the foreground, a systematic argumentation is a very useful tactic, especially if it is supplemented with personal authority and trenchancy. Exchange of experiences between the companies has also proven useful, but it is only after they have gathered their own experiences, that they believe in what others have experienced or recommend.

Inspiration can be found in [17].

Conclusion

These lessons document the more practical oriented learning from the first half of the Danish SPI initiative. Other experiences from this project are documented in a number of more traditional papers [35] - [53].

If we look at the results until now and relate these to what else is going on in Denmark within SPI, it becomes relatively clear, that external pressure gains the improvement process. A national initiative like this–where the involved companies get external support, where there is a constant follow-up, where obtained results are questioned and discussed, where mutual obligations are build between the partners, and where contractual requirements enforce a high level of commitment–constitutes and unusually fruitful and positive environment for successful SPI.

In the second half part of the project, we will focus more on effect measurements. Some measurement programmes are under implementation, and this work will be continued. In the end, such programmes are needed to establish indicators for the level of improvements achieved in a national effort like this.

Acknowledgement

We thank all participants in this large project, the four companies Brüel & Kjær A/S, Danske Data, L. M. Ericsson Denmark A/S and Systematic Software Engineering A/S. Also thank to the sponsors Ministry of Commerce (Council for Development of Business and Industry) and Ministry of Research (Centre for IT-research). Especially we thank the persons, who are co-authors of part three of the project midterm report [40], which has been the basis for this paper: Allan B. Jakobsen, DELTA Software Engineering, Ivan Aaen, Jesper Arent, Jakob Iversen, Peter Axel Nielsen, Aalborg University and Jacob Nørbjerg, Technical University of Denmark.

Literature

General literature

[1] Applegate, L. M. (1994). Managing in an Information Age: Transforming the Organization for the 1990s. IFIP Proceedings, 15-94.

[2] Arent J. & J. Iversen (1996). Development of a Method for Maturity Assessments of Software Organizations based on the Capability Maturity Model. M. Sc. Thesis. Dept. of Computer Science, Aalborg University. (In Danish).

[3] BOOTSTRAP Team (1993). BOOTSTRAP: Europe's Assessment Method, IEEE Software Vol. 10, no. 3, pp. 93-95.

[4] Brodman, J.G. & D.L. Johnson. (1995). Return on Investment (ROI) from Software Process Improvement as measured by US Industry. Software Process - Improvement and Practice, Vol. 1(Pilot Issue), pp. 35-47.

[5] Curtis, B. (1994). A mature view of the CMM. American Programmer Sep. 1994, no. 9, pp. 19-28.

[6] Daskalantonakis, M. K. (1994). Achieving Higher SEI Levels. *IEEE Software* Vol. 11, no. 4, pp. 17-24.

[7] Diaz, M. & J. Sligo. (1997). How Software Process Improvement Helped Motorola. IEEE Software, Vol. 14, no. 5, pp. 75-81.

[8] Dion, R. (1993). Process Improvement and the Corporate Balance Sheet. IEEE Software, Vol. 10, no. 4, pp. 28-35.

[9] Dunaway, D.K. & S. Masters. (1996). CMM-Based Appraisal for Internal Process Improvement (CBA IPI): Method Description. Technical report: CMU/SEI-96-TR-007. Software Engineering Institute, Pittsburgh, Pennsylvania.

[10] Emam, K.E. & L. Briand. (1997). Costs and Benefits of Software Process Improvement. ISERN-97-12. Fraunhofer - Institute for Experimental Software Engineering.

[11] Fowler, P., & S. Rifkin. (1990). Software Engineering Process Group Guide. Technical report: CMU/SEI-90-TR-24, Software Engineering Institute, Pittsburgh.

[12] Goldenson, D.R. & J.D. Herbsleb. (1995). After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success. Technical report: SEI-95-TR-009. Software Engineering Institute, Pittsburgh.

[13] Grady, R.B. (1997). Successful Software Process Improvement. Prentice Hall PTR, Upper Saddle River, New Jersey.

[14] Haase, V., R. Messnarz, G. Koch, H. J. Kugler, P. Decrinis (1994): BOOTSTRAP: Fine-Tuning Process Assessment, IEEE Software, Vol. 11, no. 4, pp. 25-35.

[15] Hayes, W. & D. Zubrow. (1995). Moving On Up: Data and Experience Doing CMM-Based Process Improvement. Technical report: CMU/SEI-95-TR-008. Software Engineering Institute, Pittsburgh, Pennsylvania.

[16] Humphrey, W. S. (1989). Managing the Software Process. Reading Massaachusetts. Addison Wesley.

[17] Jakobsen, A. B. (1998). Tricks of Bottom-Up Improvements. IEEE Software, Vol. 15, no. 1, pp. 64-68.

[18] Jonassen Hass, A. M., Johansen, J. & Andersen, O. (1997): Softwareudvikling i Elektronikindustrien. DELTA Dansk Elektronik, Lys & Akustik, D-260. (In Danish)

[19] Jones, C. (1997). Applied Software Measurement, McGraw-Hill.

[20] Kotter, J. P. (1995). “Leading Change: Why Transformation Efforts Fail.” Harvard Business Review (March-April), 59-67.

[21] Krasner, H. (1994). The Payoff for Software Process Improvement (SPI): What it is and How to get it. Software Process Newsletter, IEEE Computer Society (no. 1, September 1994), pp. 3-8.

[22] Kuvaja, P., J. Similä, L. Krzanik, A. Bicego, S. Saukkonen, & G. Koch. (1994). Software Process Assessment and Improvement - The BOOTSTRAP Approach. Blackwell.

[23] Mathiassen, L. & C. Sørensen (1996). The Capability Maturity Model and CASE. Journal of Information Systems, Vol. 6, pp. 195-208.

[24] McFeeley, B. (1996). IDEALSM: A User's Guide for Software Process Improvement. CMU/SEI-96-HB-001. Software Engineering Institute, Pittsburgh.

[25] Mintzberg, H. (1983). Structure in Fives: Designing Effective Organizations, Englewood-Cliffs, New Jersey, Prentice Hall.

[26] Nonaka, I. (1994). "A Dynamic Theory of Organizational Knowledge Creation". Organization Science, Vol. 5, no. 1, pp. 14-37.

[27] Paulk, M. C., C. V. Weber, S. M. Garcia, M. B. Chrissis (1993). The Capability Maturity model: Guidelines for Improving the Software Process. Software Engineering Institute, Addison Wesley.

[28] Paulk, M. C., W. S. Humphrey und G. Pandelios (1992). Software Process Assessments: Issues and Lessons Learned. Proceedings of ISQE92, Juran Institute, 10-11 March, pp. 4B/41-4B/58.

[29] Rout, T. P. (1995). SPICE: A Framework for Software Process Assessment. *Software Process - Improvement and Practice,* 1(Pilot Issue), pp. 57-66.

[30] Sakamoto, K., K. Kishida, N. Nakakoji (1996). "Cultural Adaptation of the CMM: A Case Study of a Software Engineering Process Group in a Japanese Manufacturing Company". Process-Centred Environments, pp. 137-154, John Wiley and Sons Ltd. New York.

[31] Senge, P. (1990). The Fifth Discipline: The Art & Practice of The Learning Organization, Century Business, London.

[32] Vinter, O., T. -M. Poulsen, J.M. Thomsen & K. Nissen (1996). The prevention of Errors through experiens-driven test efforts (PET). ESSI project number10438.

[33] Vinter, O., S. Lauesen & J. Pries-Heje (1998). A methodology for preventing requirements issues from becoming defects. (PRIDE) ESSI Project number 21167.

[34] Zahran, S. (1997). Software Process Improvement: Practical Guidelines for Business Success, Addison Wesley.

Contribution from the project

[35] Aaen, I., P. Bøttcher & L. Mathiassen (1998). The Software Factory: Contributions and Illusions. In: Proceedings of the 6th European Conference on Information Systems, Aix-en-Provence, France.

[36] Arent, J. (1998) Making Software Process Improvement Happen. Doctoral Consortium at the 6th European Conference on Information Systems, Aix-en-Provence, France.

[37] Arent. J. (1998). Making Software Process Improvement Happen: A Learning Perspective. Ph.D summer school at Magleaas, Copenhagen, Denmark.

[38] Baskerville, R. & J. Pries-Heje (1997). IT diffusion and innovation models: The conceptual domains. In: Diffusion, Transfer, and Implementation of Information Technology, T. McMaster & D. Wastell (Eds.), London: Chapman & Hall.

[39] Bøttcher, P. (1997). Comparing Total Quality Management and the Capability Maturity Model (CMM) in an Organizational Change Perspective, Proceedings of the Seventh International Conference on Software Quality, Montgomery, AL.

[40] Centre for Softwareprocesforbedring (1998). Danske Erfaringer med Forbedring af Softwareprocessen. DELTA Dansk Elektronik, Lys & Akustik, D-262. (In Danish)

[41] Falck, W., M. Gaupås, K. Kautz, A. Oppøyen & T. Vidvei (1997). Implementing Configuration Management in Very Small Enterprises. In: Proceedings of the European Conference on Software Process Improvement - SPI´97, Barcelona, Spain.

[42] Iversen, J. (1998). Data-Driven Intervention in Software Process Improvement. Doctoral Consortium at the 6th European Conference on Information Systems, Aix-en-Provence, France.

[43] Iversen, J., J. Johansen, P. A. Nielsen & J. Pries-Heje (1998). Combining Quantitative and Qualitative Assessment Methods in Software Process Improvement. In: Proceedings of the 6th European Conference on Information Systems, Aix-en-Provence, France.

[44] Iversen, J., P. A. Nielsen & J. Nørbjerg (1998). Problem Diagnosis in Software Process Improvement. Proceedings of the Twenty-first Information Systems Research Seminar in Scandinavia, Aalborg, Denmark.

[45] Jonassen Hass, A.M., J. Johansen & J. Pries-Heje (1997). BOOTSTRAP the real way to SPI. Quality Week 97 Europe.

[46] Jonassen Hass, A. M., J. Johansen & J. Pries-Heje (1998). Does ISO 9001 Increase Software Development Maturity. EuroMicro '98.

[47] Kautz, K. (1998). Even in Very Small Software Enterprises Metrics can make Sense. Proceedings of the 21st Information Systems Research Seminar in Scandinavia, Aalborg, Denmark.

[48] Kautz, K. & E. Åby Larsen (1997). Diffusion Theory and Practice: Disseminating quality management and software process improvement innovations. In: Proceedings of the 5th European Conference on Information Systems, Cork, Ireland.

[49] Lyytinen, K., L. Mathiassen & J. Ropponen (1998). Attention Shaping and Software Risk: A Categorical Analysis of Four Classical Approaches. Accepted for publication in ISR Journal.

[50] Mathiassen, L., F. Borum & J. Strandgaard Pedersen (1997). Transforming IT Management through Action Learning. Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo, Norway.

[51] Mathiassen, L. & C. Sørensen (1997): A Guide to Manage Software Engineering Technologies. In: Diffusion, Transfer, and Implementation of

Information Technology, T. McMaster & D. Wastell (Eds.), London: Chapman & Hall.

[52] Vinter, O. (1997): How to Apply Static and Dynamic Analysis in Practice. Software Quality Week 97, and Quality Week Europe 97.

[53] Vinter, O., P.-M. Poulsen, S. Lauesen, J. Pries-Heje (1997). Preventing Requirements Issues from Becoming Defects. EuroSTAR 97.

Authers and Companies

Jørn Johansen has 18 years experience in IT. He has worked for 15 years in a Danish company with embedded and application software as a developer and project manager. For the last 3 years he has worked as a consultant and registered BOOTSTRAP assessor, performing more then 25 BOOTSTRAP assessments in Denmark. Jørn Johansen is also co-ordinator the Danish SPIN-group. He is project manager in the Centre for Software Process Improvement project.

DELTA Software Engineering, Venlighedsvej 4, 2890 Hørsholm, Denmark,
E-mail: joj@delta.dk, Phone: +45-45867722, Fax: +45-45865898.

DELTA Software Engineering is a division DELTA Danish Electronics, Light & Acoustics. DELTA has been in business for more than 50 years and performs accredited testing and consultancy for discerning customers throughout Europe. DELTA has a staff of 240 and is a totally independent self-governing foundation. DELTA Software Engineering works with assessment of software products and software life-cycle processes and conducts research in these fields. Specific competence areas are software quality systems, software best practice and software product certification.

Lars Mathiassen is a professor in computer science with 23 years of experience as researcher, teacher, and consulting. His research interests focus on engineering and management of IT-systems. More particularly he has worked with project management, object-orientation, organisational development, management of IT, and the philosophy of computing. He is project manager in the Centre for Software Process Improvement project.

Aalborg University, Fredrik Bajers Vej 7E, 9220 Aalborg Ø, Denmark
E-mail: larsm@cs.auc.dk, Phone: +45-96358913, Fax: +45-98159889

Aalborg University. The Department of Computer Science at the Faculty of Technology and Science, Aalborg University, Denmark is one of the major departments of computer science in Denmark: founded in 1990, it currently has approximately 300 undergraduate/graduate students, 20 PhD students and roughly 40 employees.

SPI by IPS - Involvement, Planning, Structure

Bill Culleton
Silicon and Software Systems (S3), Ireland

Introduction

Silicon and Software Systems (S3) is a successful independent company providing silicon software and hardware design services in the areas of telecommunications, consumer electronics and computer communications etc. For 12 years we have been providing solutions to industry based on expertise, quality and dedication to meeting delivery dates. In order to build on this success it has become increasingly clear that an optimum process is essential to providing a high quality and efficient service to current and future customers. Optimising the already existing processes is however potentially an expensive task and one which is solved in many companies by setting up a dedicated group to address it. In view of the S3 culture of involvement and ownership this was not an option.

The SPI program undertaken was quite large and covered topics which ranged from improving an existing set of quality procedures through to modification of on-line process definition and support.

The improvement plan which was developed and is currently being executed has ensured that process definition and approval has involved approx. 45% of the organisation. The training methodology will ensure that more than 75% of the organisation is involved in process adaptation and approval. Apart from one dedicated co-ordinator no person has had to provide more than 40 hours of their time in the last 12 months of this part of the overall SPI project.

This paper addresses the approach taken to improve the existing quality procedures. This is felt to be the keystone to strong process definition and is essential when producing quality work for any customer. Furthermore, when used correctly it can be expected to lead to improved productivity.

The selection of an improvement model and the justification for the choice made is presented.

The paper then proceeds to discuss the improvement plan and its execution. In particular the method used to ensure maximum staff involvement and buy-in is addressed. This includes usage of pilot projects, expert teams for process definition, task forces to review their proposals and update existing procedures. It furthermore describes the training methodology adopted and highlights how this was used to increase the number of people involved in process definition and approval.

Finally a number of initial results which have been achieved are presented.

Process Background and the Drivers of Change

When S3 was founded in 1986, it was immediately decided that a quality system must be implemented. Standards and procedures were defined for the main SW development activities based on the IEEE standards. These were initially effective but as the organisation grew some problems were noticed.

Recognising the importance of controlled quality, an improvement process based on ISO9000 was embarked upon in 1993. Official accreditation was achieved in May 1994. The main outcomes of this initiative were a comprehensive library of documented procedures, both standards and guidelines as well as a very quality conscious mentality among staff.

These procedures have been in use within the company since 1994 and have certainly been seen to help meet quality criteria on projects. It had however in the meantime become increasingly obvious that they were beginning to add an expensive overhead to some smaller projects and in some cases were possibly being counter-productive.

In 1997 it was decided to tackle these issues as part of a renewed SPI initiative. There were also other issues addressed, e.g. definition of and collection of metrics, improvement of an intranet based on-line process definition and support system. These however fall outside the scope of this paper.

It was clear that there were a number of different approaches which could be taken but it was felt that the best one was very dependent on the underlying issues. As a means of identifying these a general meeting was held and individual staff were interviewed. In this way opinions were solicited from more than 30% of the staff, both development and management. The main issues identified were

- Structure of quality system library was not intuitive
- Set of procedures was often viewed as being too large for small projects
- In many areas procedures were too restrictive, in other areas they were too loose
- Many ‘standards’ were very open to interpretation and as a result much time was lost trying to interpret them for specific projects

When these issues were looked at in more detail three main points emerged

- The quality system library contained a number of very directed procedures but had no clear indication of how they fitted together. It missed any definite description of the development processes and how these individual elements fitted into it.

- Most procedures had been defined for, and evolved on large projects of many tens of man years effort. They had been made very restrictive and were now placing too much overhead on smaller projects.
- Not enough support was available in application of generic procedures to the diverse needs of projects

At about the same time that the development engineers were realising that the quality system was stagnating, senior management was beginning to realise that they did not have the visibility into how projects were being executed that they would like. Furthermore potential customers were beginning to query the level of visibility and thus the level of control.

The consensus by all levels in the organisation that these issues existed has been very important to the success of the project.

Selection of an Improvement Model

Based on the issues outlined in the previous section, it was decided that a good model must be used so that improvement could be defined and measured. After some discussion it was decided to use the Capability Maturity Model from the SEI. This has a number of characteristics which were felt to be essential to this initiative

- One of the underlying principles of the CMM is that of incremental change. Because of the size of the task and limited resources this was important
- CMM contains a predefined set of processes areas which were seen to be a good starting point. These could be used to provide the overview or holistic "end-to-end view of the process" (see [1]) which was required.
- Having a number of defined levels provides a good roadmap for continued improvement even after achievement of level 2, see "Expected Results".
- By ensuring that the development process was improved enough to achieve CMM level 2 recognition there was some measurement available. This also had the advantage of being an industrially recognised benchmark.
- The emphasis placed by the CMM on an effective support infrastructure was seen as important in terms of ensuring practical application of defined procedures as well as a means of ensuring structured improvement in the future
- It enables extra visibility into development processes for management and thus matched this management requirement. In an internationally expanding company managing this visibility correctly can help to enable world-wide management transparency.

Having identified the main issues to be addressed and the roadmap to be used to tackle these, the decision was taken to approach this initiative in the same manner as any other project in the company. This required identification of a leader responsible for development and execution of a plan as well as responsibility for the results. The plan itself is addressed in more detail in "The Project Plan" and the results are presented in "Results and Lessons Learned".

The Project Plan

In the description of the plan a number of important constraints are presented. These are felt to be typical constraints in a company of S3's size.

Plan Summary

It was decided that there were four strands to this plan. These were

- Raising process awareness and ensuring buy-in to improvements
- Process analysis, redesign and implementation
- Introduction of modified system
- Defining and providing process support

The main elements of each of these are described below

- Raising process awareness and ensuring buy-in to improvements
  - ⇒ Perform initial maturity assessment with involvement and support of developers, middle and senior management. The evaluation can be expected to provide the following results
    - ♦ it makes people aware of the issues
    - ♦ it provides a baseline figure against which improvement could be measured
    - ♦ it provides an indication of the weakest areas of the development process and thus the ones which must be addressed first

- Process analysis, redesign, implementation and testing
  - ⇒ Based on the areas identified in the evaluation described above, analyse the best methods used within the company but also throughout industry and base a standard description on these. These are then documented for use by the rest of the organisation.
  - ⇒ Existing documents can be updated using the analysis of the CMM Key Process Areas (KPA) as a starting point.
  - ⇒ Review documents using as many development personnel as possible
  - ⇒ Introduction of modified processes on pilot projects to test usefulness and practicality

- Introduction of modified system
  - ⇒ Once testing is complete the new procedures are to be released. This is to be done by a combination of announcement via internal eMail and dedicated training.

- Defining and providing process support
  - ⇒ Ensure that support in the practical application of procedures is made available to personnel and guarantee that feedback from experience gained is used to improve further. A good support structure also ensures effective deployment of best practices.

Expected Results

There were three main goals defined. In addition to these, where possible there were criteria against which their achievement could be measured. The goals and their criteria were

Goal 1	Improve quality system procedures and understanding of how they can help
Measure	No measure. Effect will only be noticed when procedures are in use.

Goal 2	Introduction of structured support organisation
Measure	Company's recognition of the need for this role and subsequent funding of it. In a design services company this is a major commitment.
Goal 3	Achievement of level 2
Measure	Is a measure in itself

Execution of the Plan

The description of execution is divided according to the four strands of the plan.

Raising Process Awareness and Ensuring Buy-in to improvements

A strong culture of quality awareness, while being something most companies would like to achieve, also has its disadvantages when trying to introduce change. These were quite apparent in S3 and provided a major challenge to the project.
Most projects are quite successful and there was a general feeling that tasks were being planned and executed effectively. This leads to a certain amount of resistance to change. This was of course tempered by the feeling that the available procedures were incomplete and thus providing potential for decreased efficiency.
In order to address this issue and to gain buy-in there were two main tasks undertaken.

- Intermediate Maturity Evaluation
  Using a fast maturity evaluation method, see [2], quick analyses were performed on the SW divisions of S3.
  From an awareness point of view, the most important feature of this method was a round-the-table session where various personnel were asked to score the various activities of each of the CMM level 2 KPAs. During the scoring session if there were strong differences of opinion on activities, the people with extreme scores were invited to offer their opinion on the activity. In this way very good discussions took place and most people left the session appreciating the need for clear process definitions and the need for an improvement program.
  As a means of ensuring that the score reflected the real situation and that all levels in the company were reached, the people invited to take part were from senior and middle management as well as development. Care was also taken to include people with a wide range of experience (1 to 17 years).
  The outcomes of this evaluation were a baseline figure and a very much increased awareness of the need for the SPI program. This counted as much for management as for developers.

- CMM Training and Workshop
  A number of experienced developers were invited to attend CMM training. Those involved were a combination of people who believed very much in process definition and control, and people who were sceptical about the idea. The workshop itself consisted of learning the CMM basics and guided tailoring of 2 KPAs for the S3 context.
  The most important result of this workshop was that the group, as a whole, decided immediately to meet another time to look at the rest of the KPAs. The

strongest advocates of this were the people who had previously been the most sceptical.

Process Analysis, Redesign, Implementation and Testing

A constraint shared by S3 and many other companies is the lack of resources due to project pressure. At the end of the day, projects paid for by customers earn money directly, an SPI project is seen as one with a return on investment which is difficult to quantify. Despite being an investment, however, this does mean that it is not possible to have a number of people working on it full time for an extended period of time. This had to be taken into account in the plan.
The basic strategy used was to plan all tasks in such a way that they were of very short duration and would not require too much time for people. In this way many people could be involved without a large impact on projects. There was only one person dedicated to the project full time, the project manager. This method of spreading work is described below.

Process analysis was performed using the workshop method described in the previous section.
The group of people involved had very wide experience on very successful and some less successful projects. By drawing on this it was possible to define processes which were based on best practices throughout the organisation and in some cases, practices encountered in previous employment.
At the end of the workshops various members of the analysis team were assigned a process. They were then responsible for documenting the results of the discussions and working the details out further with specially formed task forces. Using this method, process analysis and definition quickly involved approximately 25% of the organisation. This added to the best practice approach required as well as ensuring that a large user base had some feeling of involvement and ownership.
Having worked out the details, change requests were raised on existing standards where necessary and a number of new procedures were identified for definition.
Most of the changes to existing standards were implemented by the project manager using the change requests as the basis for work. This approach was chosen to ensure consistency as well as to reduce the burden on project teams which were under a lot of pressure at the time. When all updates were made a number of new people, as well as the authors of the change requests, were invited to review the procedures. This brought the involvement percentage to approximately 35%.

As a means of testing some of the changes being proposed it was decided to try these on pilot projects. It was agreed at the start that if any change showed even a chance of having a negative effect on a project’s quality or ability to deliver on time, it should not be used. The importance of maintaining a professional service to customers would not be compromised under any circumstances.
Two project leaders volunteered to use modified procedures. The KPAs which needed most careful testing were chosen.
One project, a digital cordless telephone application, chose to apply the Requirements Management KPA and the newly written procedures for the requirement definition phase of a new project.
The second project, a web based performance analyser chose to apply the Project Planning and Software Quality Assurance KPAs and modified procedures.
Both projects monitored the procedures effects carefully and reported regularly on

their observations. It was found that the Requirements Management and Project Planning were quite effective. It quickly emerged however, that the Software Quality Assurance process description while looking quite good on paper, was not effective. When examined in practice it was clear that it would add too much overhead to the project. It was decided not to proceed with this until such time as a redesign had been performed.
More information on these is provided in “Results and Lessons Learned”.

At this point an external consultant was contracted to examine the procedures with a view to assessing their CMM compatibility. This resulted in various changes where a number of essential aspects of the CMM had been either overlooked or misinterpreted.

After testing these KPAs and receiving the report from the external consultant, final descriptions of the KPAs were documented and procedures were updated again. The KPA descriptions are documented in a handbook which is available to all members of the SW divisions. Before releasing this everybody in the division was invited to review it. When all comments were received, involvement coverage had increased to approximately 47%.

When all procedures had been updated and released for general use a specialised training program was initiated. This is described in “Introduction of Modified Procedures”.

Introduction of Modified Procedures

Introduction followed two main paths.

All members of the SW divisions were notified by email about the release of new versions of procedures. This was known not to be an effective way of introducing the changes to people, but did ensure some awareness of the fact that changes were being made

The details of changes were introduced by organising a dedicated training course, the Software Process Training Introduction Course. This was designed with the following goals in mind

- Introduction to CMM for all members of the SW division.
- Reiteration of justification of improvement program
- Introduction of changes
- Further, guided discussion of processes, thus providing increased involvement in definition and approval
- Collection of feedback gained and updating of documents based on this

The course itself took the form of one day lecture and discussion sessions. Each process was introduced briefly by the trainer, including real examples of project problems experienced where particular aspects of processes had previously not been defined. After this the relevant documents from the quality system library were walked through and discussed. This provided all attendees with an overview of the standards as well as a further review of the procedures in question. By ensuring that people attending the course were drawn from different projects it was also possible to have a good exchange of ideas on different aspects of the processes. Any problems in the standards or good ideas from the discussions were noted on a feedback form so

that change requests could be raised to ensure further improvement.
Any suggestions received were immediately entered into an existing change request database. These were analysed and those accepted have either been implemented already or are scheduled for implementation in the near future.

To date performing reviews of documents in this manner has increased involvement in process definition and approval to approximately 80%.

Definition of and provision of process support

Documentation of procedures and training in their application of these is not sufficient. Paper gathers dust and brain cells are used for project information. For this reason a process support organisation has been defined. The main components and their responsibilities are summarised below.

Process improvement and control is managed by a Software Process Group.

A group of process experts provide advice on the application of processes and standards during the definition of project plans. These experts report regularly to the Software Process Group who then use this information to see if there are ideas which should be used elsewhere or issues which need to be addressed.

Another group of trained auditors perform regular audits on projects to verify adherence to these plans. They also report the results to the Software Process Group.

At the moment the support part of this organisation is being piloted so there are no measurable results available. Feedback from the projects involved has however been very positive so that even if nothing else the 'soft' goal of ensuring that people are happier to work with the quality procedures and system has been achieved.

Results and Lessons Learned

This chapter presents the results of this project to date as well as a number of lessons learned.
The next chapter, "KPA Analysis and Introduction", provides more details on the results of piloted KPAs. While these are not measurable they are important from the point of view of showing the effect from a customer and business perspective.

Project Goals

At this stage a number of results are available. The degree to which the established goals were met are summarised in the following table

Goal 1	Improve understanding of QS procedures and how they can help
Result	No measure, but people are already expressing satisfaction with the improvements.
Goal 2	Introduction of structured support organisation
Result	• Current project manager is now providing process application support to 3 projects. This has been approved by senior management • Division's budgets from 1998 include one person year effort per

year for this role

Goal 3	Achievement of level 2
Result	• Not yet achieved. This was originally planned for the last quarter of 1998. The results of the latest externally conducted maturity evaluation, while showing significant improvements, indicate that there are still areas which must be improved. The evaluation itself indicated an average improvement of 44% improvement in performance.

Other Results

Apart from achievement of the project goals there were some other results achieved. One very important one has been that one of the pilot projects was so successful that the customer has approached S3 with a proposal for further work. This is based very much on the quality of the processes used on the pilot project.

The following chapter, see "KPA Analysis and Introduction", provides more information on the piloting of processes and the level of success achieved

Lessons Learned

- Treatment of this sort of initiative as a project, following the same principles as any other is the most effective way to achieve results. Having a solid plan enabled the management of resources and constantly changing availability.
- Use at least one or if possible a number of external consultants and use them early. They can provide an objective view of the organisation.
- Use a mix of experienced and inexperienced people for process definition. The experienced people may have good ideas but the inexperienced people often have very good questions and can force practicality onto working methods.
- Keep the business goals in mind. Even at an early stage in such an improvement project potential results can be seen.

KPA Analysis and Introduction

One very useful observation early on in the analysis phase was that the standard KPA definitions provided by the SEI mapped very closely to the way of working in S3. This was very useful in that process descriptions have been prepared by describing most aspects of the KPAs in terms of S3 while adhering strictly to the SEI structure. This has made analysis and subsequent verification far simpler than would otherwise have been the case.

The following two sections, "Project Planning" and "Requirements Management", present results of two KPAs which were piloted.

Project Planning

Improvements were introduced to the planning process including the documenting of the plans on a web based performance analysis tool for IBM.

This involved significant changes to the amount of detail which must be considered at an early stage of the planning process as well as the manner in which this is documented, e.g. more detail on and tabulated presentation of risk-analysis. The increased amount of detail and improved clarity of the documents produced was felt by S3 management to provide improved visibility into the project thus improving the ability to manage and track the project significantly. For IBM management it provided similar visibility and thus increased their confidence in S3.

The increased level of detail provided also had a major benefit for the project team members. By settling details early in the project lifecycle there were fewer ambiguities later on and less time was spent on discussions of topics which would previously have been left open. Though not measured this would clearly have had a direct impact on productivity. To a lesser degree, but not insignificant, it also had an effect on the quality of the product.

The project itself resulted in a product which was delivered on time and was accepted on initial release. Furthermore S3 has been approached by other departments within IBM with a view to further work based on the reputation gained by the success of this one.

Requirements Management

Changes in the manner in which initial requirements were gathered and documented were introduced when starting a project for Telital.

Initial specification was originally required in a very short time scale. While it took longer than was required, this had been identified early and had been discussed and agreed to by Telital. Without a clear process here this negotiation would have been more difficult for both parties.

With respect to the requirements themselves, these were documented in such a manner that an early review showed that they were well understood and the customer was happy to accept them as the basis for development. For the project team members, they were confident that they understood the customer requirements and had a good basis to plan and execute the project.

The result of the project has been a DECT handset which was delivered on time and has recently achieved type approval.

Within S3 it has been decided to use this method to define the features of S3's own proprietary product to ensure easier comparison with future customer requirements.

The Author

Bill Culleton, B.A., B.A.I., M.Sc, M.I.E.I is currently project manager for the S3 software process improvement project as well as being chairman of the S3 Software Process Group.
He has six years experience in research and development for Computer Aided Design applications for IC layout and design. This included both user support and application promotion, thus providing a strong insight into the practical effects of errors in the systems. The effects include both problems for users of the system as well as the commercial effects.

He has six years experience in the development of GSM Base Station Operation and Maintenance software. This time has included active involvement in every aspect of the project lifecycle starting at initial requirement specification through to management of field testing and delivery to end users. More than half the time has been spent as team and project leader within S3 as well as acting as project manager with responsibility for managing the development and testing activities in the four companies involved in the project. The activities have provided further insight into the effect of the development process on the success of projects in term of meeting budgets as well as strict quality criteria.

The Company, Silicon and Software Systems (S3)

S3 was founded in 1986 in Dublin, Ireland. Since its foundation, S3 has worked with many of the world's leading electronics multinationals, designing state-of-the-art Integrated Circuits, Embedded Software and Hardware Systems for the merging Communications, Computing and Consumer Electronics markets.
An international company, with a presence in Europe, USA and The Far East, the company consists of approximately 300 employees (88% of whom are engineering staff) who offer expertise in designing solutions which combine the speed of silicon and the flexibility of software.
Also offering solutions in both disciplines separately, some of the company's recent achievements have included the design of one of the largest silicon chips in the world to date and the development of advanced telecommunications software for a variety of DECT products for both voice and data applications.

Acknowledgements

The author would like to acknowledge the following persons and companies for their valuable support and advice during the execution of this improvement project.

Martin Farnan, SW director in S3. Sponsor of the project, mentor and provider of pick-me-up motivational conversations on a regular basis.

Declan Kelly of S3. My predecessor, responsible for laying the groundwork for the project and engaging in useful, thought provoking discussions throughout its lifetime to date.

Fran O' Hara of Insight Consulting, Ireland. External consultant who in addition to providing CMM advice furnished a lot of practical advice specific to the S3 context.

Sami Zahran of IBM. External consultant who in his training workshop provided a useful insight into CMM but possibly more important, insight into how the principle of SPI can be taught in an enthusiastic manner.

IBM and Telital. Allowed me to use the examples of KPAs tested on projects executed for them. Looking at the success of these projects it would appear that this co-operation has resulted in a win-win situation.

References

[1] Zahran, S., Software Process Improvement, Practical Guidelines for Business Success.

[2] The Interim Maturity Evaluation Toolkit. Provided by “Origin TA/IPS, Veldhoven, The Netherlands”

Experiences from practical software process improvement

Seija Komi-Sirviö
Markku Oivo
Veikko Seppänen

VTT Electronics, P.O. BOX 1100, FIN-90571 OULU, FINLAND
Seija.Komi-Sirvio@vtt.fi, Markku.Oivo@vtt.fi, Veikko.Seppanen@vtt.fi
fax: +358 8 551 2320, tel. +358 8 551 2111

Abstract: This paper describes a systematic multi-paradigm approach to software process improvement called Pr²imer (Practical Process Improvement for Embedded Real-time Software). Pr²imer has been developed by VTT Electronics for analysing and improving embedded software development processes. It has been used in industrial software process improvement programmes carried out both for SMEs and large multi-national firms. It integrates software process analysis, goal-setting, improvement planning, measurement and piloting into a total quality management framework. Among other methods, Pr²imer utilises a Goal/Question/Metric (GQM) method developed by the University of Maryland. GQM is a top-down, goal-oriented and measurement-based quality improvement method. In this paper, the usability of Pr²imer will be evaluated. Furthermore, a discussion focusing on the key success factors for process improvement programmes is presented. The discussion is based on results gained from almost twenty industrial improvement projects during the past five years.

Keywords: **software process improvement, measurement, embedded software**

Introduction

The volume of embedded software in electronic products is constantly growing. In addition to hardware and mechanics, software has become one of the core product technologies. Typical examples of embedded computer systems are telecommunication products, industrial process control systems and electronic instruments. Customer-specific versions of electronic products are often implemented by using embedded software. Therefore, the quality and management of the software process has become a critical success factor within the electronics industry. An immature software development process may result in poor product quality. In addition, the management and predictability of an obscure software process is very insecure.
Section one of this paper introduces Pr^{2}imer (Practical Process Improvement for Embedded Real-Time Software) method. Section two describes experiences gained from practical process improvement. Section three summarises the results and presents guidelines for further work.

Practical Process Improvement for Embedded Real-time Software – PR^{2}IMER

Pr^{2}imer is a practical and systematic approach for improving the quality of software development process [1] [2]. It provides an overall improvement framework for integrating and selecting software analysis, measurement, modelling and improvement methods in order to meet the quality improvement requirements set by the company. Pr^{2}imer includes the following four activities for process improvement (see Fig.sks. 1)

1. quantitative and/or qualitative analysis concerning the current state of the process and the product,
2. definition of the target state of the software process with measurable process and product goals,
3. plan for practical process improvement activities, and
4. pilot operation and commissioning.

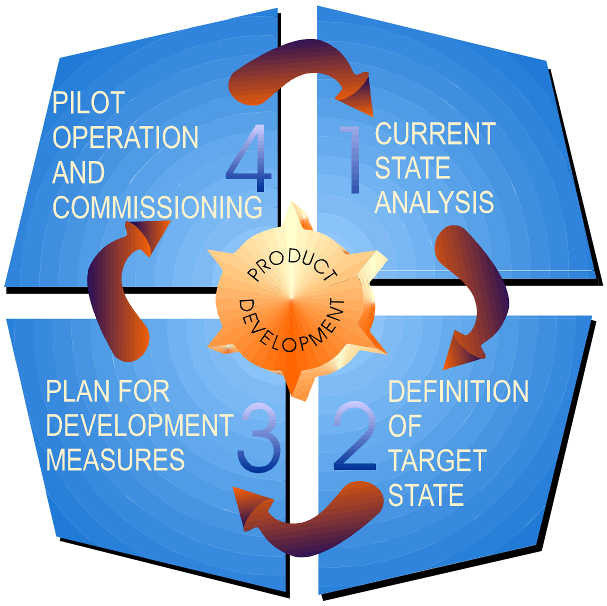

PILOT OPERATION AND COMMISSIONING
4
1
CURRENT STATE ANALYSIS
PRODUCT DEVELOPMENT
PLAN FOR DEVELOPMENT MEASURES
3
2
DEFINITION OF TARGET STATE

Fig.sks. 1. Software process improvement activities supported by Pr^{2}imer.

Pr^{2}imer utilises different methods and techniques which are selected according to the situation and needs of the company. In all phases and activities of Pr^{2}imer, close co-operation between the software development projects and the improvement team is emphasised. From the initial planning of the improvement work onwards, interaction between the ongoing software development projects and the planned process improvement activities is crucial. Process improvement goals and steps have to be adapted for the schedules of the software development projects. In addition, the quality requirements for process improvement must conform with the goals planned by the project organisation. Before release for large-scale use, it is necessary to perform pilot projects in which the improved practices are carried out according to the improvement plan.

Current state analysis

Analysing the current status of the software process or subprocess forms the basis for any improvement initiatives. The purpose of the analysis is to describe and evaluate the current software development practices and to identify problems or bottlenecks that are possible subjects for improvement.

Pr^{2}imer utilises two analysis strategies: qualitative and quantitative. The qualitative analysis technique includes a semi-structured interview, which is usually performed by using an analysis framework originally developed in the Esprit project AMES [3]. Qualitative analysis produces descriptive process models which include descriptions of the organisational context and application domain, software development and management practices. Qualitative analysis also provides evaluations concerning the software development methods and tools that are used in projects. Furthermore, the supporting techniques such as software development guidelines, templates, quality assurance etc. are evaluated. The most serious problems, as well as opportunities for improvement, are identified. Identification is based on evaluating the results and by taking into consideration the goals of the company as well as those of the project. There are several different techniques for analysing the current status. The most suitable one is selected according to the needs and situation of the company. Quantitative analysis can be performed by using for example such well-known questionnaire-based software process assessment methods as BOOTSTRAP [4], CMM [5] [6], Trillium [7] or ISO15504 (also known as SPICE) [8] which provide information concerning the maturity level of the process. If a purely measurement-based approach is utilised in Pr^{2}imer, the analysis can be performed by using a GQM (Goal/Question/Metric) [9] method. In this case, Pr^{2}imer approach is very similar to the QIP paradigm [10].

Definition of target state

The definition of the target state takes place after a mutual understanding of the current situation and the improvement areas is reached. The second phase of Pr^{2}imer consists of improvement goal setting and prescriptive process modelling. This phase also includes measurement planning which is done according to the GQM method.

Development areas are analysed and evaluated by taking into consideration the objectives set by both the organisation and software product development. Normally, the analysis produces numerous different observations that are prioritised managers and project members who will pilot the improved practices. When the improvement goal or goals are set, updates to process models, practices, methods and use of tools are defined in order to support the achievement of the goal. The concrete outcome of this definition may result in new guidelines for software development, document templates, work instructions, et cetera.

Pr^{2}imer uses measurements in monitoring the success or failure of the improvement

activities and in providing immediate guidance and feedback for the project members. Pr^{2}imer utilises a goal oriented Goal/Question/Metric (GQM) method. The GQM approach is based on the assumption that in order to provide meaningful measurements, an organisation must first specify the measurement goals in both organisational and project levels. Following this, measurement goals have to be traced back to the data that defines the goals operationally. Finally, a framework for interpreting the data with respect to the stated goals must be defined.
GQM is a hierarchical model (Fig.sks. 2) starting with a measurement goal which is defined according to a template with five dimensions expressing

- the object of measurement: *which software engineering objects are measured*?
- the purpose of measurement: *why are these objects measured?*
- the quality focus*: which properties of the objects are measured?*
- the subject of measurement (viewpoint): *who is interested in these measurements?*
- the context of measurement: *in what environment are the objects measured?*

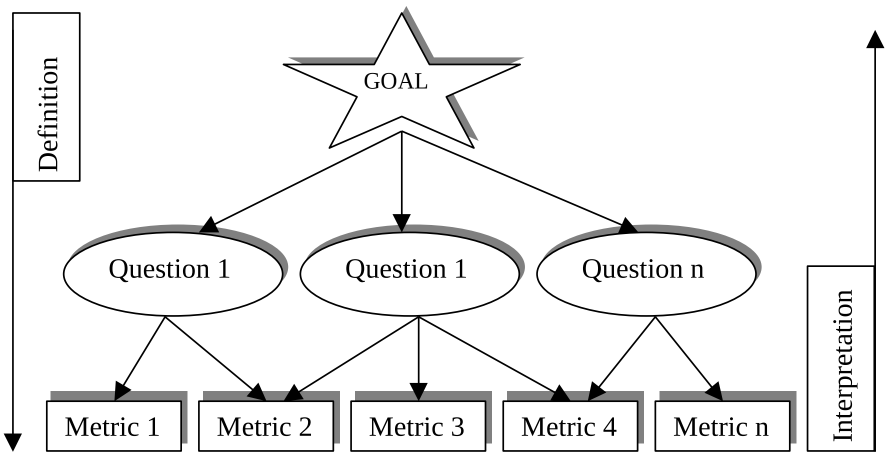


Fig.sks. 2. The GQM approach.

The measurement *goal* is refined into a set of *questions*, which characterise the object with respect to chosen issues from a selected viewpoint. The questions are then defined by a set of *metrics* in order to provide quantitative answers. One metric may contribute to different questions, and one question is typically answered by taking into account several metrics.

Plan for development activities

The third phase of Pr^{2}imer consists of planning improvement and measurement activities or, in other words, planning process improvement implementation. The main results are concrete process improvement and measurement plans. A step-by-step procedure plan is needed, even if process improvement initiatives have been thoroughly targeted, from the very beginning, to a specific area of software development. When planning the process improvement steps, the main question is how to proceed from the current situation towards the new practices defined in phase two. To answer this question, the improvement goals and strategy for proceeding are described, improvement initiatives are scheduled, the organisational and project level follow-up are planned and the training demands are clarified. In addition to the overall improvement plan, a measurement plan according to the GQM method is developed. This plan contains a detailed procedure for metric collection. It describes who is responsible for collecting the data, and when and how it is to be collected. In summary, the third phase of Pr^{2}imer deals with implementation and measurements: in what way are the improvement initiatives implemented and how are the measurements to be utilised in pilot projects.

Pilot operation and commissioning

Before taking the revised software process or subprocesses into large-scale use, new practices must be evaluated by testing them in a pilot project or projects. Alongside the pilot project, new practices are evaluated in feedback sessions where project members analyse the collected measurement data. This way, the current status of piloting is identified and possible corrective actions can be taken during piloting. The feedback is bi-directional, which means that if necessary, experiences from the pilot projects are utilised in revising the process improvement plans.
After piloting, the success of improvement actions will be evaluated mainly according to the GQM plan. Informal feedback from the pilot projects is taken into consideration as well. After the piloting based on the analysis and experiences, it is decided how the revisited process will be utilised in product development and what are the further requirements for improvement.

Experiences from practical process improvement

During the past five years, Pr^{2}imer method has been utilised in more than twenty process improvement cases, with fifteen different companies. The results have been good. In the following, a discussion concerning the experiences of using the Pr^{2}imer method in process improvement is presented.

A cluster of companies where Pr^{2}imer has been applied

The domain of applications for Pr^{2}imer usage varies from consumer electronics to safety critical medical instruments. However, most of the cases belong to the telecommunication sector. We have co-operated with large international companies such as ABB, Datex, Nokia, and Valmet, as well as with small and medium size companies such as Polar Electro and X-Net.
In practice, the more mature the software development process of the company, the more specific the focus of improvement. This feature affects the methods that are selected in different phases of Pr^{2}imer, particularly during the current state analysis. We have practical experiences concerning focused improvement of the testing process, which in many cases has been identified as the most problematic subprocess. Other demanding processes are change management and requirements management. In eight cases, the whole software process has been the object of improvement initiatives.
In the following, both general observations and remarks specific to Pr^{2}imer phase are discussed.

The importance of the top management commitment

Before the process improvement can truly start, various negotiations and initial planning activities are crucial for the success of any process improvement initiatives. It is essential that the right persons are involved in and committed to the improvement work from the very beginning. This includes both persons owning the process and persons managing the resources that are needed to execute the improvement work. If the pilot project is not given enough resources for instance personnel to participate in the improvement planning phase, and if there is no time for learning and utilising the new practices in the project, problems are bound to appear. Experience has shown that the pilot project (team) is often very enthusiastic and committed to improvement work in the beginning, but the revised practices can be abandoned since the time pressure in the project is severe. The software development work is hurriedly completed, in a similar manner as before, with the remark "we know this is not a good way to proceed but it is the fastest one at the moment".

Software development projects have seldom enough time allocated to fulfilling the requirements set to development. Learning and piloting new practices usually causes additional work for the pilot project. If this is not supported by providing additional resources to the project, problems are to be expected in the later phases of improvement.

Experiences of the Pr^{2}imer phases

<u>Current State Analysis</u>

The purpose of current state analysis is to find out both the weaknesses and the possible improvement areas. Current state analysis can be thorough, covering the whole software development process, or it can be limited to include only one subprocess such as requirements analysis. It is recommendable to start with a full analysis in order to get a good overview of all processes, especially if the process improvement initiatives are the first of their kind in a company. An extensive analysis will clearly indicate which processes the improvement actions could be targeted at. Without a large scale analysis, the decision concerning improvement areas may be based on assumptions only. It is therefore possible that the process focus for improvement is not the most urgent one, which in its turn may cause results that are less than optimal.
Current state analysis can be performed by using quantitative methods such as CMM or BOOTSTRAP. Alternatively, it can be managed with a qualitative method which uses different techniques focusing on describing the actual software development process, methods and tools. According to our experience, these methods support and complement each other. Quantitative methods evaluate processes and provide a numerical assessment which indicates the maturity of the processes by comparing them against a standard process model. Later, when revised practices have been taken in full use in the company, the process can be checked with re-assessment in order to verify whether the desired improvement occurred from the maturity point of view. So far, one weakness pertaining to the most commonly used assessment methods has appeared: namely, these methods do not provide adequate support for analysis concerning the special features of embedded software development. After all, quantitative method alone is not sufficient if the actual software development process is not identified - and quite often it does not correspond to the official process described in a quality manual or other document. Descriptive process modelling forms the basis for any accurate improvement suggestions. In those cases where we have used only qualitative methods for defining the actual practices, the result of analysis has offered adequate framework for further actions. Fig.sks.3 presents a high level example result of current state analysis of requirements engineering process.
The current state analysis is the least effort consuming phase for the company, if analysis is to be conducted by external experts, as was the case in all these examples. According to our experiences, the calendar time needed for current state analysis varies from a few weeks to one and a half month. Normally the current state analysis has not revealed weaknesses that are completely unknown [12]. One important benefit of the analysis has turned out be the following: the situation is documented and mutual understanding is achieved, which strengthens the commitment of project members and the company and forms a basis for defining goals for new practices.

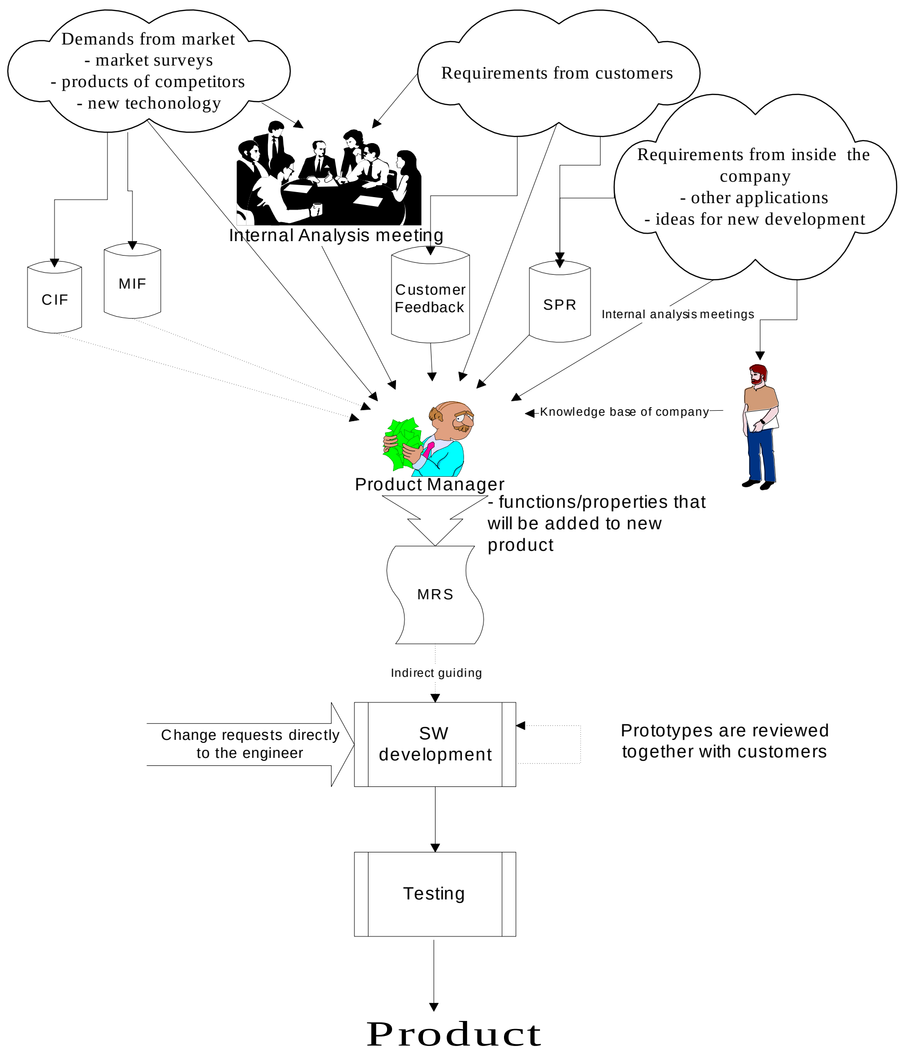

Demands from market
- market surveys
- products of competitors
- new techonology
Requirements from customers
Requirements from inside the company
- other applications
- ideas for new development
Internal Analysis meeting
CIF
MIF
Customer Feedback
SPR
Internal analysis meetings
Knowledge base of company
Product Manager
- functions/properties that will be added to new product
MRS
Indirect guiding
Change requests directly to the engineer
SW development
Prototypes are reviewed together with customers
Testing
Product

Fig.sks.3. A high-level example result of current state analysis.

<u>Definition of Target State</u>

Definition of the target state consists of three main activities: improvement goal setting, detailed definition of improved practices and measurement planning.
The most used technique for improvement goal setting is brainstorming session managed by improvement experts. A brainstorming technique has sometimes been criticised for producing too many improvement goals, but in general, it has been found useful. New practices are designed to meet the improvement goals that have been decided in the beginning of the target state definition. This includes prescriptive process modelling which leads to concrete results such as software development guidelines or instructions.
Inviting persons with different positions (i.e. product managers, quality managers, project managers, engineers etc.) to these meetings has proved to be very useful. This way, all viewpoints can be taken into account. In normal industrial software development environment, it is difficult to bring together people that represent different viewpoints of software development, mainly due to scheduling problems. Invitations to selected pilot project or projects are important in order to strengthen their commitment to the improvement work. In Scandinavian countries, this might be a crucial issue since making decisions is, above all, a democratic undertaking. Fig.sks. 4 shows an example of the revised process for testing. It was defined in co-operation with a company in Pr^{2}imer

phase 2.

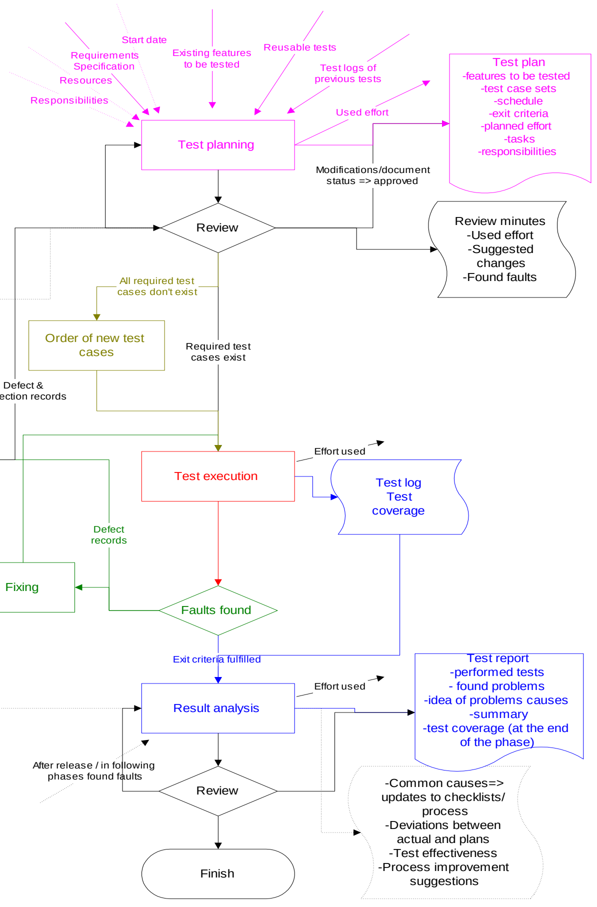


Fig.sks. 4. An example of the proposal for the testing process.

Pr^{2}imer emphasises the role of measurements in process or product improvement. Measurement data provides the best mean for following the improvement actions and managing the software development projects. We have successfully applied the Goal/Question/Metric method in two ways: in both following and managing the improvement activities and in defining the current state of a certain process. In Table.sks. 1 an extract from a GQM plan is presented. In this case, the improvement was targeted at the testing process, and the current metrics values are identified by the analysed defect database. This GQM plan is defined to monitor whether the improvement initiatives function to improve the testing process as planned.

Table.sks. 1. An example of a GQM plan.

GOAL1:	
Analyse	***the product and process***
from the point of view of	***developer/tester/manager***
in order to	***improve it***
with respect to	***reliability***
in the context of	***G-Company/QM-Project***
Q.1.7 What is the amount of defects found in testing?	
	M.1.7.1 For each test step/phase: Total # of defects found
	For each defect found:

	M.1.7.2 Severity (fatal, minor, cosmetic)
	M.1.7.3 Finding activity (module testing, integration testing, system testing)
	M.1.7.4 Date of find
Q.1.9	*What is the defect lifetime?*
	For each defect found:
	M.1.9.1 Origin (phase when made: requirements specification, specification, design, implementation, testing)
	M.1.9.2 Finding activity (document review, code review, module testing, integration testing, system testing)
	M.1.9.3 Find phase (phase when found: requirements specification, specification, design, implementation, module testing, integration testing, system testing, customer use)
	M.1.9.4 Date of find (M.1.7.4)
	M.1.9.5 Fixing date (M.1.7.5)

Starting various improvement actions at once is ill-advised. Regardless of how much this precaution is stressed, there seems to be a human desire to prove that in this particular case, it might be possible to proceed in a wider front. In the beginning of software process improvement, it is deemed more important to get good results from a narrow area than to only get some results in a wider area. The GQM method itself strongly emphasises the role of the project in which the measurements are implemented in practice.

Depending on the target process or processes, the definition of the target state requires approximately one to two months' worth of calendar time. During this phase, contribution from the company is of crucial importance.

Plan for Development Activities

The third phase of Pr^{2}imer concentrates on implementing the improvement actions. The main results of this phase are a measurement plan and an improvement plan. During this phase, the steps required for moving from the current situation to the target situation are planned. In addition, the measurements for supporting the improvement initiatives are determined. The improvement plan unites the current state analysis, improvement goals, defined new practices and planned measurements; furthermore, it defines the strategy which is required to proceed in practice. While planning the improvement steps, the characteristics and the schedule of the pilot project must be taken into account. These two factors have to be extended to cover the overall schedule of improvements. In addition, the need for training is also determined. If necessary, it is still possible to fine-tune the improvement initiatives during this phase.

Measurement-related activities such as measurement data presentation, data collection and analysis will increase the amount of work in project management. In order to diminish the work load, VTT Electronics has developed a PC-based tool called MetriFlame. MetriFlame is utilised in automatic measurement data collection, analysis and presentation [12] [13]. It supports the GQM method by providing full GQM planning and analysis support. MetriFlame has been used primarily in automating (as extensively as possible) the collection and analysis of measurement data. In addition, it has been applied to support GQM approach where the metrics required are tailored from project to project. Fig.sks. 5 presents the MetriFlame measurement environment. Features supporting the connection to different databases and output data formats are also presented.

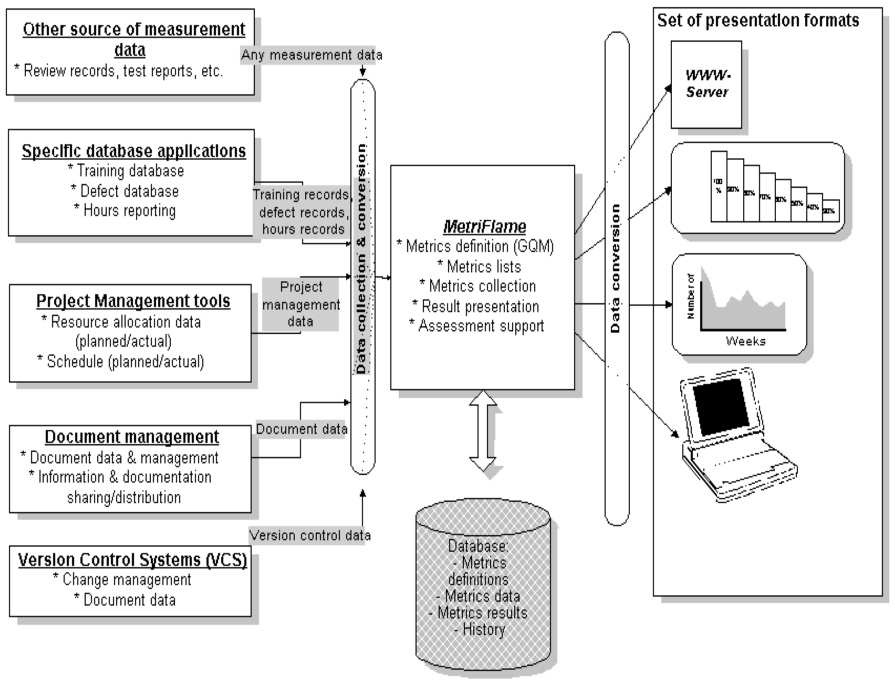

Other source of measurement data
* Review records, test reports, etc.
Any measurement data
Specific database applications
* Training database
* Defect database
* Hours reporting
Training records, defect records, hours records
Project Management tools
* Resource allocation data (planned/actual)
* Schedule (planned/actual)
Project management data
Document management
* Document data & management
* Information & documentation sharing/distribution
Document data
Version Control Systems (VCS)
* Change management
* Document data
Version control data
Data collection & conversion
MetriFlame
* Metrics definition (GQM)
* Metrics lists
* Metrics collection
* Result presentation
* Assessment support
Database:
- Metrics definitions
- Metrics data
- Metrics results
- History
Data conversion
Set of presentation formats
WWW-Server
Number of
Weeks

Fig.sks. 5. MetriFlame tool environment [12].

<u>Pilot Operation and Commissioning</u>

With reference to calendar time, piloting is the most time consuming phase. The projects we have been involved in have lasted a minimum of one year. During this phase, giving continuous support and feedback to the pilot project is extremely important, as already reported in [14]. The status of improvement actions has to be checked on regular basis (for example monthly or bimonthly). This is done in joint feedback sessions with the pilot project. In these sessions, the measurement results are analysed by the project members who know the exact circumstances concerning the acquisition of the measurement data. The pilot project provides a realible interpretation of the measurement results. The graph in Fig.sks. 6 is an example created by using MetriFlame and the defect database.

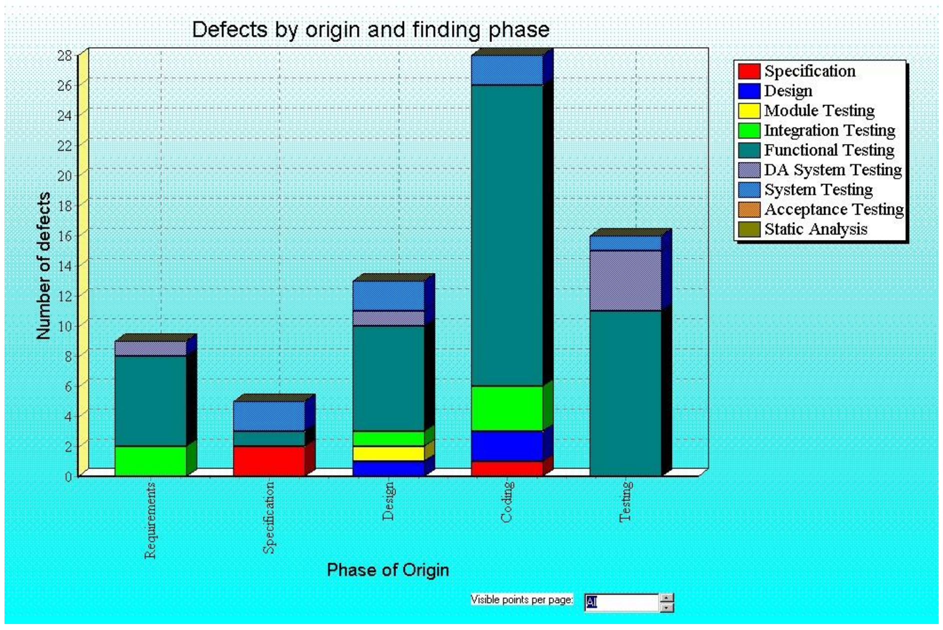


Fig.sks. 6. An example result of MetriFlame used in real feedback session.

During piloting, it is necessary to provide practical support in form of feedback sessions [15]. If necessary, the improvement plan is updated. Prescriptive process models, as well as other plans such as GQM and measurement plans, may require similar updating. After the new practices have been successfully piloted in the pilot project or projects, they will be put into full-scale use in the organisation. This phase has proved to be quite demanding and requires thorough planning to proceed smoothly. After the new practices are in full-scale use, a new Pr^{2}imer improvement circle can be initiated. Process improvement is a challenging undertaking, due primarily to the constant progress and evolvement associated with it.

Conclusions and further work

In this paper, we have introduced the main principles behind a practical process improvement method called Pr^{2}imer. In addition, we have described various examples and experiences achieved during the past five years. Pr^{2}imer divides process improvement into four main phases, starting with a current state analysis and continuing with an improvement goal setting and target state definition. Before piloting activities, improvement implementation and improved practices are planned in such a way that the project in question can make full-scale use of them. In the domain of embedded software, the demand for software process improvement is constantly growing due to the increasing volume of software in embedded products. In supporting management and improvement initiatives, the already important function carried by measurement has received additional emphasis during the recent years. Even though experiences and results concerning the use of Pr^{2}imer improvement method have been very favourable, we are constantly developing Pr^{2}imer in order to fully satisfy the companies' demands. In addition to automatic measurement tool support, the next enhancement will introduce a methodology which enables linkage between product quality characteristics and the influencing process. This methodology is currently under development in an Esprit project called PROFES (PROduct Focused improvement of Embedded Software processes) [16] [17] and it will supplement embedded software development with completely new possibilities [18].

References

[1] Mäkäräinen, M., Komi-Sirviö, S., *Practical process improvement for embedded real-time software,* Proceedings of the 5th European Conference on Software Quality, Dublin, IR, 16 - 19 Sept. 1996. (1996), 408 - 416

[2] Karjalainen, J., Mäkäräinen, M., Komi-Sirviö, S., Seppänen, V. Practical process improvement for embedded real-time software. *Quality Engineering,* Vol. 8, no 4, 1996.

[3] Application Management Environments and Support, Esprit 3 Project 8156, Technical Annex, Version 1.0, August 1993.

[4] P. Kuvaja, J. Similä, L. Krzanik, A. Bicego, S. Saukkonen, G. Koch. "Software Process Assessment & Improvement – The BOOTSTRAP Approach", Blackwell Publishers, 1994.

[5] Humphrey, W. S.: Managing the Software Process. SEI Series in Software Engineering. ISBN 0-201-18095-2, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[6] Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, Pittsburgh, Pennsylvania, 1993.

[7] Telecom Product Development Process Capability, Version 2.3d, Bell Canada, 1993.

[8] ISO/IEC TR 15504-2: *Information Technology – Software Process Assessment – Part 2: A Reference Model for Processes and Process Capability*". Technical Report type 2, International Organisation for Standardisation (Ed.), Case Postale 56, CH-1211 Geneva, Switzerland, 1998.

[9] Basili, V. R., Caldiera, G. & Rombach, H., D. The Goal Question Metric Approach. In Marciniak, J. J. *Encyclopedia of Software Engineering.* Wiley, 1994.

[10] Victor R. Basili, Gianluigi Caldiera. *Improve Software Quality by Reusing Knowledge and Experience*. Sloan Management Review, pp. 55-64, Fall, 1995

[11] van Latum, F., van Solingen, R., Oivo, M., Hoisl, B., Rombach, D., Ruhe, G., Adopting GQM-based measurement in an industrial environment*, in IEEE Software,* January/February. Vol. 15 (1998) Nr: 1, pp. 78 – 86

[12] Parviainen P., Komi-Sirviö S., Sandelin T., Measurement-based improvement of critical software subprocesses: Experiences from two industrial cases, in*: Proceedings of the Conference of Software Process Improvement, SPI'98*, Monte Carlo, December 1998

[13] Parviainen, P., Järvinen, J. & Sandelin, T. Practical Experiences of Tool Support in a GQM-based Measurement Programme. In Hawkings, C., Ross, M., Staples, G. & Wickberg H. *INSPIRE II, Process Improvement, Training and Teaching for the Future.* The British Computer Society, 1997. Also in Software Quality Journal, Vol. 6, No. 4, 1997.

[14] van Latum, F., Oivo, M., Hoisl, B., Ruhe, G., 1996, *No Improvement Without Feedback: Experiences from Goal-Oriented Measurement at Schlumberger.*

[15] Parviainen, P., Järvinen, J. & Sandelin, T. *An example model of GQM -feedback loop.* ROIHU -Project Report, VTT Electronics, 1997.

[16] PROFES: ESPRIT project no. 23239: PROduct Focused improvement of Embedded Software processes. URL: http://www.ele.vtt.fi/profes/

[17] Andreas Birk, Janne Järvinen, Seija Komi-Sirviö, Pasi Kuvaja, Markku Oivo, Dietmar Pfahl. "PROFES – A Product-Driven Process Improvement Methodology". In *Proceedings of the Fourth Conference on Software Process Improvement (SPI '98),* Monte Carlo, Monaco, December 1998.

[18] Dirk Hamann, Janne Järvinen, Andreas Birk, and Dietmar Pfahl. "A Product-Process Dependency Definition Method". In *Proceedings of the 24th*

EUROMICRO Conference: Workshop on Software Process and Product Improvement, volume II, pp. 898-904, IEEE Computer Society Press, 1988.

CURRICULUM VITAES

Seija Komi-Sirviö works as a research scientist at VTT Electronics' Software Engineering Group. She was born in 1966. In 1992, she received her Master's Degree in Information Processing Science from the University of Oulu, Finland. She has worked as a System Analyst at CCC Companies from 1989 to 1994. From 1994 onwards, she has carried out research concerning the metrics of software process improvement in VTT Electronics' applied research projects. Her research interests include software process assessment and improvement, as well as GQM based software process measurement. She has published several papers in international conferences.

Markku Oivo is the Chief Research Scientist and Head of Software Engineering Group at VTT Electronics, where he has worked since 1986. He is responsible for initiating and managing both applied research projects and industrial development projects for a broad range of clients in software engineering. His fields of interest include software engineering, software process improvement and measurement, production of embedded software, object-oriented methods, as well as quality assurance and improvement. He has previously worked at the University of Oulu 1981-82, at Kone Co. in 1982-86, and he has held visiting positions at the University of Maryland in 1990-91 and Schlumberger Ltd. in 1994-95. He also holds a docentship in software engineering at the University of Oulu. He has published approximately 50 papers in international conferences and journals. Markku Oivo received his MSc and PhD from the University of Oulu. He is a member of the IEEE and ACM.

Veikko Seppänen was born in 1958. He earned his MSc, LicTech and Dtech degrees in software engineering in 1983, 1985 and 1990, respectively, from the University of Oulu, Finland. He worked as a software engineer at Nokia Data in 1982-83 and as a research scientist and section head at VTT Computer Technology Laboratory in 1983-93. From 1994 onwards, he has worked as a research manager at VTT Electronics and from 1995 onwards, as a research professor of embedded software. He also holds a docentship at the University of Oulu. In 1986-87, he was a non-degree graduate student at the University of California at Irvine, USA and in 1991-1993, a JSPS Postdoctoral Fellow at the University of Kyoto, Japan. Veikko Seppänen has published over 70 scientific and professional papers concerning embedded software engineering methods, tools and solutions.

VTT ELECTRONICS

VTT is the largest contract R&D organisation in the Nordic countries. The number of personnel is approximately 2600. VTT's main functions include technology transfer and consultancy projects with the industry. VTT is divided into nine units (including VTT Electronics) which are independent business units within VTT. The annual turnover of VTT is estimated to grow from 167 MECU (1995) to 193 MECU by the year 2000.

VTT Electronics, one of nine units of VTT, employs 250 experienced professionals. It offers its services to all manufacturers of products containing electronic parts. VTT Electronics' basic business aim is to improve the competitiveness and profitability of the industry by

- accelerating the integration of information technology and electronics into products, and developing new applications of electronic products,
- ensuring the prospected growth of the electronics industry by offering effective R&D services
- supporting the emergence of new industries by producing new technologies and innovations.

VTT Electronics R&D services are used by electronics, telecommunications, process automation, mechanical engineering, and instrumentation industries. VTT Electronics offers its clients the following services:

- contract research and development for industrial clients,
- design, development, and prototyping of electronic products,

consultancy in methods and process development, technology transfer.

Session 6 – Object Oriented SPI

Impact on Introducing Object Oriented Software Development Methodologies

Paul Sullivan
ESBI Computing Ltd., Dublin
Pat Caffrey
ESBI Computing Ltd., Dublin

The Rational Objectory Process - A UML-based Software Engineering Process

Presenter: Sten Jacobson
Rational Software Scandinavia AB

O.O.S.I. OBJECT ORIENTED SYSTEM INTEGRATION PROJECT N. 10987

CARICCHIA PAOLO
AEROPORTI di ROMA

Impact on Introducing Object Oriented Software Development Methodologies

Paul Sullivan
ESBI Computing Ltd., Dublin
Pat Caffrey
ESBI Computing Ltd., Dublin

1.0 Introduction

The SCOOP project objective was to enable an holistic view of the impact of introducing OO software development methodologies and tools. The specific objectives of the project were:

- Selection of an OO methodology.
- Selection of software development tools incorporating the chosen methodology.
- Production of a test piece of OO software.
- A three stage assessment of the test software production experience, i.e. a direct productivity comparison, examination of the impact of OO on the whole baseline project (Stores Controller), and examination of the impact on the whole company.

The SCOOP project had a number of deliverables, both internal and public, which will reflect the success of the work achieved during the 8-month duration.

2.0 Selection of OO methodology

To find as many OO methodologies in the marketplace, the Internet was used to find a list of books containing OO methodologies.

The book “A comparison of Object Oriented Methodologies” was used as a guideline to selecting the following methodologies in more detail:

2.1 Fusion

The Fusion method is a combination of different sections of different methods. It was discounted almost immediately due to its failure to describe an organised methodology for developing applications. A large amount of documentation is produced during the Fusion methodology, however the processes by which that documentation is produced, the manner in which that documentation links - or its overall cohesiveness, and the actual worth of the documentation produced is sadly lacking.

2.2 OMT

Object- Oriented Modelling and Design
Prentice Hall International 1991

OMT along with Booch is considered to be one of the best Object-Oriented methodologies. It is used extensively by many companies, has a wealth of documentation available and a large number of case tools support it. The text describing OMT is excellent with a section on Analysis, which is worth reading regardless of the design methodology to be used. The main difficulty with OMT is not what is produced, but the diagrams used to represent it. The diagrams in this methodology are angular and to the uninitiated (even with a notation guide) are difficult to follow; being ambiguous until a textual description is read. Where OMT fails miserably is when it comes to design, as it lacks the step-by-step approach of the analysis phase.

2.3 UML

The Unified Method V 0.8
The Unified Modelling Language V 0.91

The Unified Modelling Language (UML) is a combination of Grady Boochs’ Booch methodology and Rumbaughs’ OMT methodology. It was initially developed by Grady Booch and James Rumbaugh, both of whom now work for Rational Software Ltd. Ivar Jacobson then joined Rational and thence the UML team.

The fact that there is a definite similarity of approach and thinking between the Booch and OMT methodologies is apparent when comparing the two methods. This feeling is backed by the following remark:

In comparing its’ self with the Booch-91 methodology the OMT manual states:

“The similarities between the approaches are more striking than the differences, and both approaches complement one another”.

Unfortunately UML was currently in development and as such was not considered a contender for selection.

2.4 Selected Methodology - Booch

Object-Oriented Analysis and Design With Applications ~ Second Edition (1994)

The Booch design methodology is like OMT extensively used has ample documentation and support tools. It has been chosen over OMT primarily because it deals not only with the analysis stage of a project but also the design. The diagramming notation used in Booch is also more readily accessible and easier to understand than other methodologies. "Booch's notations are very comprehensive and can be used to document almost any aspect of the system.". One of the advantages of Booch is the fact that it is extremely versatile and robust. The diagramming notation can as stated above be used to represent almost any feature of a given system.

It is felt that the OMT methodology offers an extremely good process concerning the analysis section, and for this reason while Booch shall be the methodology used, procedural and process methods concerning the analysis of a problem may be taken from OMT. As UML develops further it may then be possible to move over from Booch with a flavouring of OMT to UML.

3.0 Selection of OO Development Tools.

To find as many OO development tools in the market, the Internet was used again to find companies and their OO products. Questionnaires were sent out to these companies to verify the suitability of their products to use in the SCOOP project. The twenty-eight questionnaires received were split up into the following categories : CASE, Development Tools, Object Request Brokers(ORB) and Object Orientated Databases (OODBMS). Products falling into the category of either ORB or OODBMS were discarded as being unsuitable to our business. The remaining tools were scored using a weighing system described below :

Heading	Case	Dev	Heading	Case	Dev
Multi-user development Environment	0.8	1	Available Support	0.8	0.8
Platforms Targeted	0.6	1	Available Training	0.8	0.8
Customisability	0.9	0.6	Road Map	0.5	0.5
Code Generation	1	0.4	Performance	0	1
OO techniques Supported	2	2	Learning Curve	0.9	0.8
Licensing	1	1	Hardware Requirements	0.5	0.8
Client Base	0.7	0.5	Evaluation Software	0.8	0.7
Inter-operability	0.3	1	Reporting Tool	0.9	0.2

Migrate from VB/VC++	0.7	0.4	Support Tools	0.6	0.6

A score of 0-5 was given under each heading and the weighing applied. This scoring system was then applied to each of the remaining tools giving the following results:

CASE

Product	Description	Score	Reason
Forte	High-end s/w modelling and development environment for 3-tier distributed applications		Prohibitive cost
ObjectMaker	Flexible CASE tool		Unacceptable level of marketing and support
ObjectTeam	CASE tool heavily centered around Informix with an OO front-end		Limited package with unsuitable focus
Paradigm Plus	OO component modelling tool for VB,C++, Delphi, etc.	67.1	Licensing, Inter-operability, platforms targeted, evaluation software
Rational Rose	**OO CASE for VB, C++, etc. tightly integrated with Microsoft**	**72.3**	**Accepted**
Select CASE	***OO CASE tool for VB, C++, Delphi***	***79.6***	***Accepted***
System Architect	***PowerSoft code partner OO CASE tool***	***82.3***	***Accepted***

Development Environment

Product	Description	Score	Reason
Arranger	IEF-based companion to Composer		Costing and parent company
Borland C++	32-bit C++ environment		Unacceptable level of marketing and support
Composer	IEF-based companion to Arranger		Costing and parent company
Delphi	***Borland OO GUI environment for Object Pascal***	***81.3***	***Accepted***
Elements	Distributed n-tier OO application		Prohibitive cost of multiple products
Forte	High-end s/w modelling and development environment for 3-tier distributed applications		Prohibitive cost
MS Foxpro 5.0	Microsoft Xbase OO client/server		Limiting programming language
MS Visual	Microsoft Java in MS	71.7	Immature technology with

J++	development studio environment		possibly unstable and unsupported ports
Sun Java	SPARCworks GUI workshop		Unsuitable development platform and unstable/unsupported platforms
MS VB 4.0	***First generation of OO VB***	***78.1***	***Accepted***
MS Visual C++	***Microsoft C++ in development studio environment***	***83.7***	***Accepted***
Obsydian	Mid-end s/w modelling, development, partitioning tool		Prohibitive cost
OpenROAD	OO 4-GL across heterogeneous platforms		Run-time licensing, no OLE or ODBC, minimal GUI to cater for multiple platforms
Optima++	PowerSoft C++ GUI environment	52.2	Support tools, inter-operability, client base, road map, learning curve
PowerBuilder 5.0	PowerSoft 4GL	76.6	Licensing, compilation/distribution, available support
Visual Age	IBM smalltalk environment		Unacceptable level of marketing and support
Visual Objects	OO 4GL		Replaced by OpenROAD

After scoring the CASE and OO development tools , it was decided that one CASE tool and 2 Development tools would be selected. Evaluation software was bought for the following tools:

Category	**Product**	**Score**	**Evaluation**
CASE	System Architect	82.3	Very confusing interface. Functionally - "Jack of all trades and a master of none". Rejected.
	Select CASE	79.6	GUI not intuitive but much better than System Architect. Too centered around the OMT methodology. Rejected.
	Rational Rose	**72.3**	**Very user friendly provided good documentation and supported Booch methodology. Accepted.**
Development Environment	**Delphi**	**81.3**	**Excellent Visual component library – performs better than Visual Basic. Has most OO features. Accepted.**
	MS Visual Basic 4.0	78.1	Shorter learning curve than Delphi but not enough OO features. Rejected.
	MS Visual J++	71.7	Immature technology. Rejected
	MS Visual C++	**83.7**	**All OO features, performs very well. However very poor at screen painting. Accepted.**

4.0 Training

All the members of SCOOP (1 Project Manager, 1 Team Leader and 2

Analysts / Programmers) were given an excellent 4 day training course in the Booch OO methodology presented by Rational Software. A 3 day training course was organised for Delphi in Ireland , however this training course was poorly designed and was not as useful as expected.

5.0 Selecting a Stores Controller module.

In selecting a Stores Controller (SC) module the following criteria was used:

- The implementation time of the module must approximately match 30 days.
- The chosen module must not have many database tables or links to other modules. The module must be as independent as possible and have good metrics.

Also the following implementation assumption was made:

As Stores Controller was implemented in 1994 on Windows 3.1, certain improvements can be made to the user interface, as the SCOOP implementation will be developed in Windows 95. Also the new concepts of three-tier architecture and OLE may be used to implement SCOOP's version of the SC module. Even though the end result may look different the same functionality will be emulated.

Using the above criteria and implementation it was decided that the Location functionality in SC would be selected.

6.0 The OO implementation Experience.

While implementing the Location functionality of Stores Controller the following related concepts were added to the SCOOP implementation: -

- Design Patterns : These are standard set of design problems and their solutions. Using both the Internet and a book called "Design Patterns - Elements of reusable Object-Oriented Software". This helped in the design phase.
- Application Partitioning : It was decided to have a very flexible architecture by partitioning the SCOOP application into GUI classes, controller classes, Business classes and the Database classes.
- Iterative Life Cycle : This was based on eliminating the project risks early on in the life cycle.

7.0 Assessments

7.1 Metric Assessment

The purpose of this assessment was to compare the original module metrics with the SCOOP metrics to calculate which method was more productive. Before the comparison was made the learning curve was taken out of the SCOOP metrics to try and compare like with like. The following metrics were calculated: -

Module	Design	Coding	Test	Total (days)	Lines of code
Existing System	5	20	7	32	4630
OO System	20	9	6	35	3343

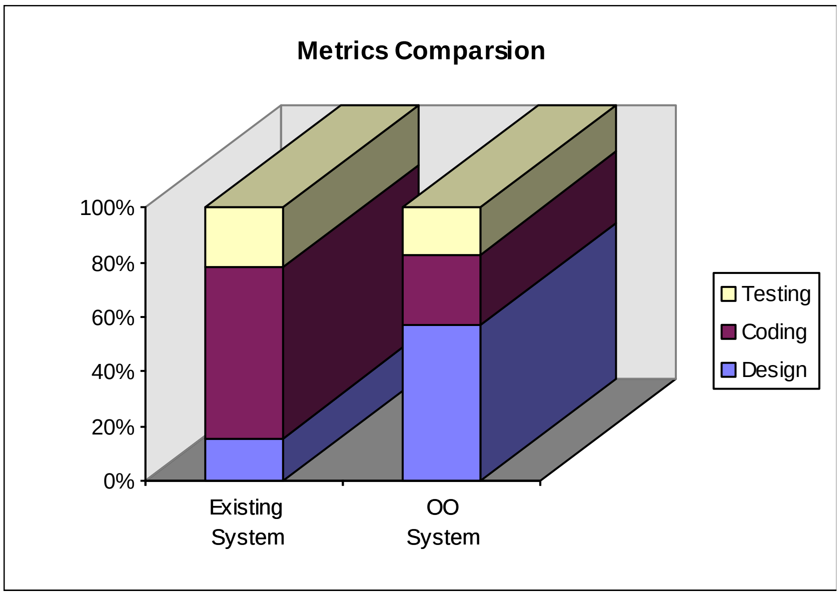


Fig 1 PSULL.1

Based on the metrics the following conclusions about using the OO methodology were made :

- There was substantially more time spent designing in the OO methodology and less time coding for the following reasons
  a) The design using OO techniques is a much more thorough process. All problems even implementation issues must be thought out at this stage. Also if a business function is left out or is added at a later stage , the class design may change radically. The designer must also have the 'big picture' view of the project and must known how the business area is used throughout the system.

b) There is much more documentation in the design phase. There are class diagrams, scenario diagrams, use cases and Axis of change documents. In the existing software process method there is at most two documents.

c) *Coding takes less time due as the design documentation provides classes that can be grouped together into programmable packages. These packages can be written in isolation and accessed through interfaces.*

- The OO Metrics should decrease with time. If the 'big picture' view is taken in the OO design phase and if the system is well designed , classes can be reused. Therefore as the life cycle progresses more classes are written and re-use becomes more realistic. *In this scenario the percentage re-use could be estimated at 60% i.e. 60% of the code could be potentially re-used in the next module in the Stores Controller application and also for another application.*
- The OO metrics should decrease in the maintenance phase. The OO approach abstracts functionality better than more traditional approaches.

7.2 Product Assessment

The purpose of this document was to assess the impact in applying the OO methodology to the entire Stores Controller product. In order to implement SC using the Booch methodology the following roles must be created and training provided for these new roles:

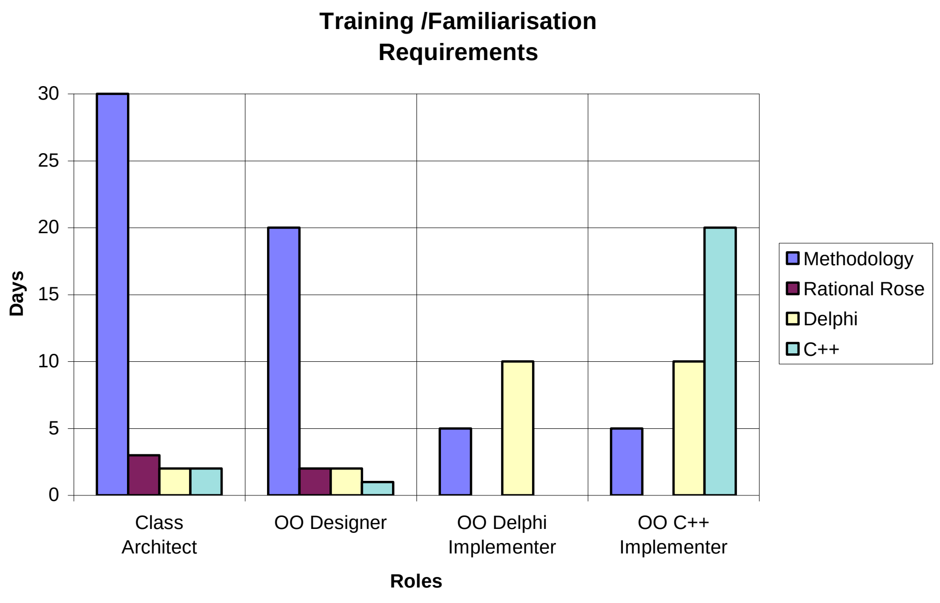


Fig 2 PSULL.2

It was calculated that the 225 man-days were required to train the required staff to implement Stores Controller effectively.
To estimate the number of man days needed to implement Stores Controller, was achieved by getting the actual time spent implementing each SC phase and multiplying these actuals by a ratio calculate by the SCOOP experience. The following table shows how much time is spent in each phase when uses the OO methodology in relation to the existing software process.

Phase	**OO Methodology**
Planning	125% (1.25 times)
Analysis & Design	140%. (1.4 times)
Coding	60%. (0.6 times)
Testing	90%. (0.9 times)

Using the above table and the actual metrics of the Stores Controller system the following comparison is made:

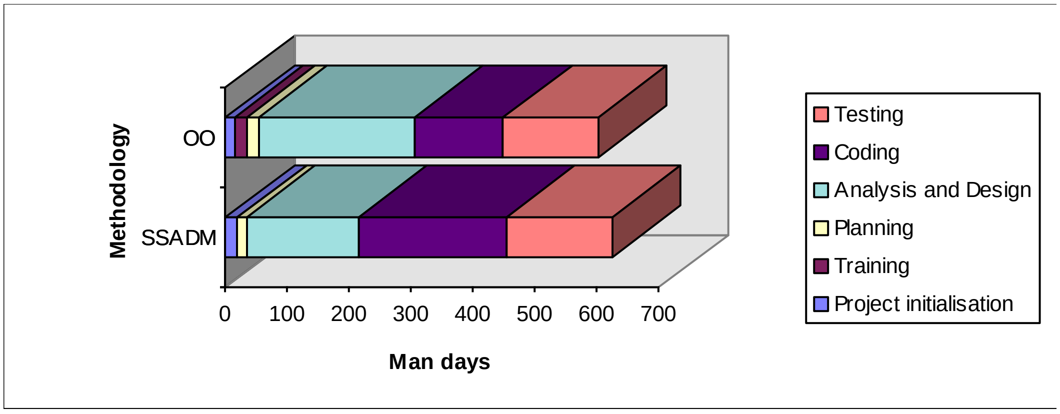


Fig 3 PSULL.3

Note: The large training overhead was not included in the above comparison.

7.3 Company Assessment

This assessment describes the impact on adopting the OO methodology on the entire company. The following impacts were found:

- Staff. The current staff will need to be trained in the new OO methodology. Staff roles will have to change. These roles are described below :
  - **Class architect**. The chief designer for each project who is familiar with the entire functionality of the project.
  - **OO Designer.** A designer very familiar with the OO methodology.
  - **OO Delphi programmer**. A programmer would

understand OO concepts and can use Delphi to implement these concepts.
- **OO C++ programmer**. A specialist programmer to write 'C++' using OLE.
- **Class Librarian**. A person who is familiar with all the business and utility classes used in the company. This person will be responsible for object re-use across projects.
- **Technical Architect**. This person understands the technical framework and the development environment on which the projects are built.
- **OO Project Manager**. A person who knows how to manage a successful OO project.

- Life Cycle. The company life cycle could change dramatically if the iterative development process is introduced to deal with the complex problem of OO project management. The Quality life cycle will have to change to reflect the correct use of the OO methodology.

8.0 Problems Encountered

The following problems were encountered during the SCOOP project:

- Inadequate training for Borland Delphi.
- During the project one of the personnel had to leave the project.
- The change from a traditional Top-Down approach to an Object-Oriented approach took a long time to master.
- Many of the development tools vendors did not answer the questionnaire.

9.0 Conclusions and Recommendations.

The major disadvantage in adopting the OO methodology is the overhead or learning curve. The staff will have to be given formal training in the OO methodology. The company life cycle may also change and there may be an overhead involved in changing the Quality documentation. Also the managing of OO projects will be more difficult.

However SCOOP feels that the OO methodology is the correct way forward for the following reasons

- The time to market will decrease over time.
- The framework devised by SCOOP has greater flexibility and solves many of the problems with the existing architecture. The OO methodology made this possible.
- Allows greater abstraction and object re-use.
- Puts more emphasis on design, which means more system bugs are

caught at design time.

These advantages will be value for money over time, as the following graph will illustrate:

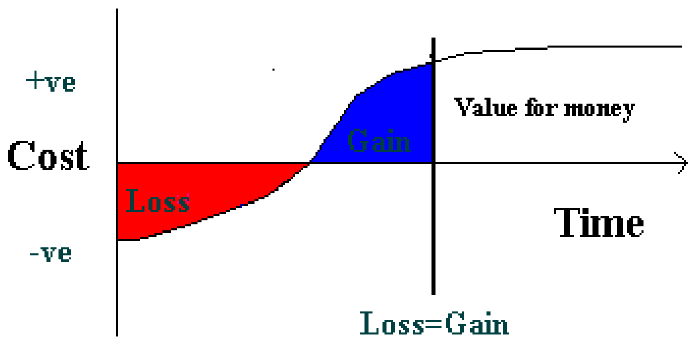


Fig 4 PSULL.4

The following are the recommendations of SCOOP to improve the software process in ESBI Computing: -

- Investigate and review the current SCOOP architecture and enhance the current SCOOP framework as appropriate. Also review Delphi 3.0 , Visual J++ ..etc. and develop a clear future road map for MC/SC technical architecture.
- Produce a demonstration system of the SCOOP architecture to get user feedback. Based on the user feedback update the SCOOP framework with agreed changes.
- Review and select a module of the MC/SC suite of application for re-engineering using the SCOOP framework and methodology.
- Establish a training strategy to match both the short term implementation of the selected module and also address the longer term goals.
- Create a project plan for the selected module and set up standards for the iterative development life cycle.
- Review Quality system and implement required changes to adopt an iterative life cycle and an OO methodology.

Disclaimer

The opinions stated in this document are purely these of ESBI Computing and relate to the findings of the SCOOP project.

Acknowledgement

ESBI Computing wish to thank the European Commission for its assistance throughout the SCOOP project, without, which the SCOOP

project, would not have been such a success.

References

References

[1] *Object-Oriented Development The Fusion Method*, Prentice Hall 1994. ISBN 0-13-101040-9

[2] *Object-Oriented Analysis and Design With Applications, Second Edition (1994)*, Addison-Wesley Publishing.

[3] *Object- Oriented Modeling and Design*, Prentice Hall International 1991.

[4] *The Unified Method V 0.8 The Unified Modeling Language V 0.91*, Public domain - Internet.

[5] *A Comparison of Object-Oriented Methodologies*, The Object Agency 1995.

[6] *Object Oriented Methods*, Ian Graham Addison Wesley.

APPENDIX A - Company Background

ESBI Computing (ESBIC) is a member of the ESB International group of companies, which is a subsidiary of ESB (Electricity Supply Board) Ireland. Established in 1989, ESBIC has built up a impressive track record by delivering information technology solutions to the international utility market.

ESBIC has ISO 9001 certification and employs a Structured System Analysis and Design Methodology (SSADM) approach to the development life cycle. The legacy product that the SCOOP project used as a baseline, is called Stores Controller (SC), and was developed using client-server architecture. SC is a large part of the Maintenance Controller (MC) product suite. The development tools of Visual Basic 3.0 / 4.0,Visual C++ version 5.0 and an Oracle Database were employed in the original software.

APPENDIX B - Author Information

Author 1

Paul Sullivan

Qualifications

BSc in Computer Applications
Dublin City University

IT Experience

- 5-6 years IT experience from developer to team leader
- Microsoft Certified Professional in Visual Basic
- I am well practised in the use of Visual Basic, Delphi, Visual C++, Oracle and NT.
- I have adhered to the quality standards of ISO 9001 for over 2 years.
- The last five years I have worked on the re-engineering of a large utility-based maintenance management system. It is now a client server application running in many power stations around the world.

Author 2

Pat Caffrey

Qualifications

BE
University College Dublin

IT Experience

- 15 years IT experience from developer to team leader
- Research and Development Manger for ESBIC
- Production Manager for Stores Controller

APPENDIX C – Design Differences between SSADM and OO.

1.0 Documentation.

SSADM	**Object Oriented**
Requirement document	Use Cases
Module Summary	Class diagram
Screen Design	Screen Design
Detailed Module Design	Class Diagram
Database Modeling	Database Modeling
	Axis of change
	Scenario diagrams

2.0 Differences

The use case diagram in the OO methodology has many levels. The first level is a simple definition of the business function the use case is encapsulating. When the class diagram is finished the relevant classes can be associated to the use case. When the Database Modeling is complete the relevant entities added. When the scenario diagrams are completed the relevant scenarios can be attached. The use case is the link from the original requirement to the analysis and design – this does not occur as easily with SSADM.

The module summary, in SSADM, which is a textual description of what the module does and how it interfaces with other modules, is generally pure text. The actual function definitions only get added in detailed design (in another document). However in the class diagram the analysis phase has all the classes and their relationship to each other, the design phases brings this class diagram a stage further by explain HOW they interact with each other – by defining the methods and properties. This is done using a notation

that is much closer to how the code will work.
Below is a sample use case and class diagram: -

Define a Location

Functional Details

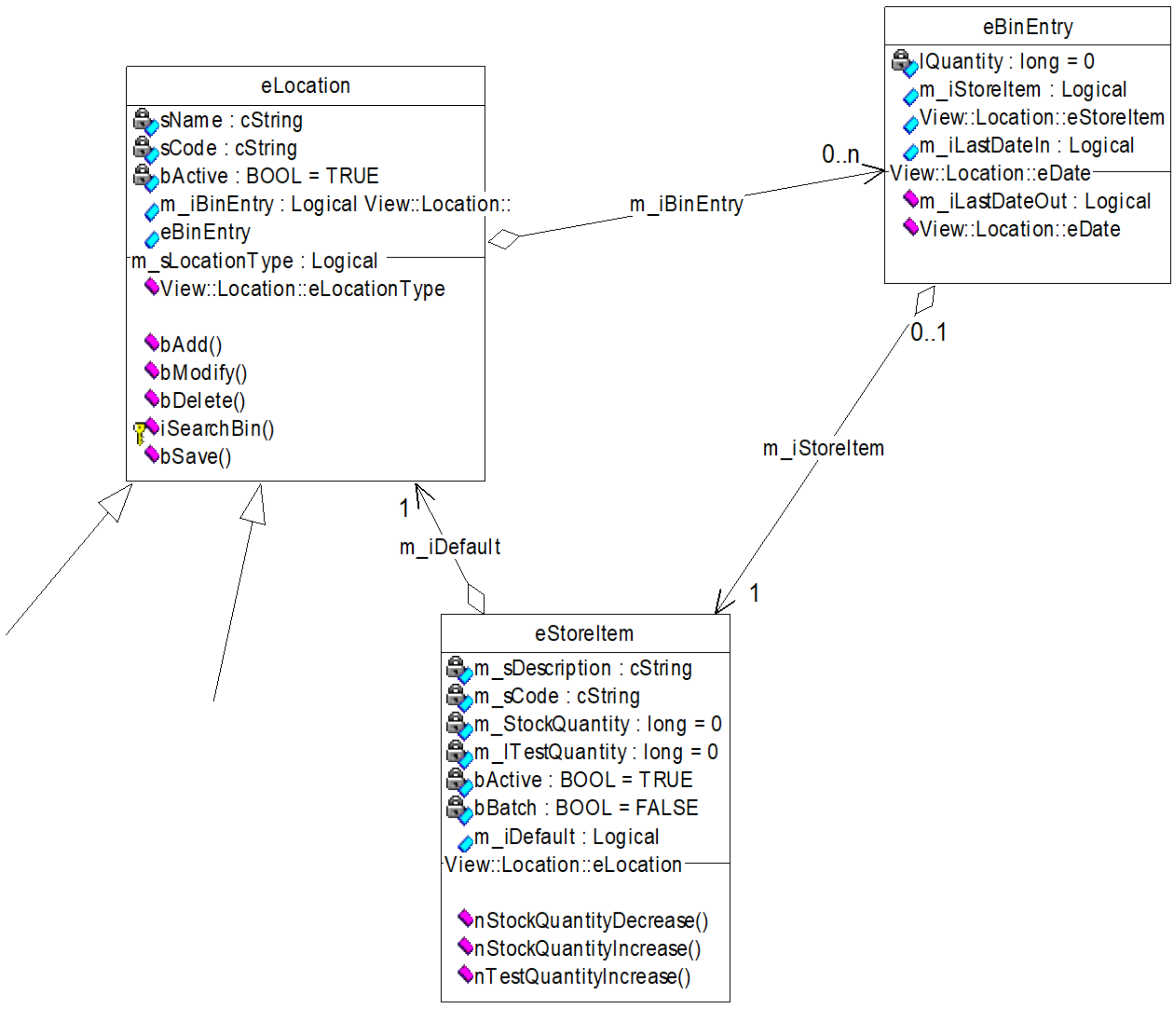
Author	Diarmuid Mac Carthy
	Power stations generally have a warehouse on-site. These are typically organised into areas of physical storage such as floor space, shelves, trolleys, pallets, bins, drums, etc. The majority of these will be normal storage areas but some may be dedicated to inspecting suspect materials (e.g. a safe area to accommodate hazardous goods), while others may be used for routine testing (e.g. a workshop for stress testing). Locations occasionally become inactive, usually because of structural defects or building renovations, contamination or cleaning. This use case provides a mechanism for the storeman to create a new area of storage. In a situation where the new system replaces an older one, or where a previous manual system is in operation, a means must be provided for reflecting the fact that quantities of store items are already in stock at a particular location. This use case provides a mechanism to allow the storeman specify the store items and their quantities that already exist at the new location.
Actors	StoresMan
Pre-Conditions	• The new location must have a unique identification
Post-Conditions	• A uniquely identified location has been created

Implementation Details

Classes Used	eLocation eLocationType eBinEntry eStoreItem

GUI Details	•
Database Implementation	• Location is inserted into the database through stored procedure *SP_INS_LOCATION* with parameters location code, description location type, active flag, comments, row version .

Class Diagram (Sample)


eLocation
sName : cString
sCode : cString
bActive : BOOL = TRUE
m_iBinEntry : Logical View::Location::eBinEntry
m_sLocationType : Logical View::Location::eLocationType
bAdd()
bModify()
bDelete()
iSearchBin()
bSave()
eBinEntry
lQuantity : long = 0
m_iStoreItem : Logical View::Location::eStoreItem
m_iLastDateIn : Logical View::Location::eDate
m_iLastDateOut : Logical View::Location::eDate
0..n
m_iBinEntry
0..1
m_iStoreItem
1
1
m_iDefault
eStoreItem
m_sDescription : cString
m_sCode : cString
m_StockQuantity : long = 0
m_lTestQuantity : long = 0
bActive : BOOL = TRUE
bBatch : BOOL = FALSE
m_iDefault : Logical View::Location::eLocation
nStockQuantityDecrease()
nStockQuantityIncrease()
nTestQuantityIncrease()

The Rational Objectory Process - A UML-based Software Engineering Process

Presenter: Sten Jacobson
Rational Software Scandinavia AB

Abstract

This paper presents an overview of the Rational Objectory Process. The Rational Objectory Process is a full lifecycle software engineering process bringing Unified Modeling Language (UML) best practices to the fingertips of each software developer. Objectory is a controlled iterative process, with strong focus on architecture. It is a use-case driven, object-oriented process, using the UML as a notation for its models. Objectory can be configured to fit a wide range of projects.

Introduction—What is Objectory?

The Rational Objectory Process is a Software Engineering Process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software, meeting the needs of its end-users, within a predictable schedule and budget. The Objectory process captures many of the best practices in modern software development in a form that is tailorable for a wide range of projects and organizations.

Objectory is an *iterative* process. Given today's sophisticated software systems, it is not possible to sequentially first define the entire problem, design the entire solution, build the software and then test the product at the end. An iterative approach is required that allows an increasing understanding of the problem through successive refinements, and to incrementally grow an effective solution over multiple iterations. An iterative approach gives better flexibility in accommodating new requirements or tactical changes in business objectives, and allows the project to identify and resolve risks earlier. [1, 2]

Objectory is a *controlled* process. This iterative approach is only possible however through very careful *requirements management* and *change control*, to ensure at every point in time a common understanding of the expected set of functionality and the expected level of quality, and to allow a better control of the associated costs and schedules.

Objectory activities create and maintain *models*. Rather than focusing on the production of large amount of paper documents, Objectory emphasizes the development and maintenance of *models*—semantically rich representations of the software system under development. [3, 7, 8]

Objectory focuses on early development and baselining of a robust software *architecture*, which facilitates parallel development, minimizes rework, increases reusability and maintainability. This architecture is used to plan and manage the development around the use of software *components*.

Objectory development activities are driven by *use cases*. The notions of use case and scenarios drive the process flow from requirements capture through testing, and provides coherent and traceable threads through both the development and the delivered system. [7]

Objectory supports *object-oriented techniques*. Several of the models are object-oriented models, based on the concepts of objects, classes and associations between them. These models, like many other technical artifacts, use the Unified Modeling Language (UML) as the common notation. [4]

Objectory supports *component-based software development*. Components are non trivial modules, subsystems that fulfill a clear function, and that can be assembled in a well-defined architecture, either ad hoc, or some component infrastructure such as the Internet, CORBA, COM, for which an industry of reusable components is emerging. [5]

Objectory is a *configurable* process. No single process is suitable for all software development. Objectory fits small development teams as well as large development organization. Objectory is founded on a simple and clear process architecture that provides commonality across a family of processes and yet can be varied to accommodate different situations. It contains guidance on how to configure the process to suit the needs of a given organization.

Objectory encourages objective on-going *quality* control. Quality assessment is built into the process, in all activities, involving all participants, using objective measurements and criteria, and not treated as an afterthought or a separate activity performed by a separate group.

Objectory is supported by *tools*, which automate large parts of the process. They are used to create and maintain the various artifacts—models in particular—of the software engineering process: visual modeling, programming, testing, etc. They are invaluable in supporting all the bookkeeping associated with the change management as well as the configuration management that accompanies each iteration.

Process Overview

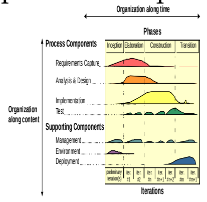
Two Dimensions

The Rational Objectory Process can be described in two dimensions:

- along *time*, the life-cycle aspects of the process as it will unroll itself
- along *process components*, which groups activities logically by nature

The first dimension represents the *dynamic* aspect of the process, as it is enacted, and is expressed in terms of cycles, phases, iterations and milestones.

The second dimension is represents the *static* aspect of the process: how it is described in terms of process components, activities, workflows, artifacts, and workers.



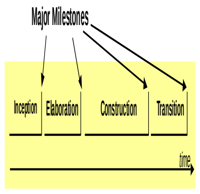
Phases and Iterations

This is the dynamic organization of the process along time.

The software lifecycle is broken into *cycles*, each cycle working on a new generation of the product. The Objectory process divides one development cycle in four consecutive *phases* [10]

- Inception phase
- Elaboration phase
- Construction phase
- Transition phase

Each phase is concluded with a well-defined *milestone*—a point in time at which certain critical decisions must be made, and therefore key goals must have been achieved [2].



The phases and major milestones in the process.

Each phase has a specific purpose.

Inception Phase

During the inception phase, you establish the business case for the system and delimit the project scope. To accomplish this you must identify all external entities with which the system will interact (actors) and define the nature of this interaction at a high-level. This involves identifying all use cases and describing a few significant ones. The business case includes success criteria, risk assessment, and estimate of the resources needed, and a phase plan showing dates of major milestones.
At the end of the inception phase, you examine the lifecycle objectives of the project and decide whether or not to proceed with the development.

Elaboration Phase

The goals of the elaboration phase are to analyze the problem domain, establish a sound architectural foundation, develop the project plan and eliminate the highest risk elements of the project. Architectural decisions must be made with an understanding of the whole system. This implies that you describe most of the use cases and take into account some of the constraints: non functional requirements. To verify the architecture, you implement a system that demonstrate the architectural choices and executes significant use cases.
At the end of the elaboration phase, you examine the detailed system objectives and scope, the choice of an architecture, and the resolution of major risks.

Construction phase

During the construction phase, you iteratively and incrementally develop a complete product that is ready to transition to its user community. This implies describing the remaining use case, fleshing out the design, and completing the implementation and test of the software.
At the end of the construction phase, you decide if the software, the sites, the users are all ready to go operational.

Transition phase

During the transition phase you transition the software to the user community. Once the product has been put in the hand of the end users, issues often arise that require additional development to adjust the system, correct some undetected problems, or finish some of the features that may have been postponed. This phase typically starts with a “beta release” of the systems.
At the end of the transition phase you decide whether the lifecycle objectives have been met, and possibly if you should start another development cycle. This is also a point where you wrap up some of the lessons learned on this project to improve the process.

Iterations

Each phase in the Objectory process can be further broken down into iterations. An *iteration* is a complete development loop resulting in a release (internal or external) of an executable product, a subset of the final product under development, which grows incrementally from iteration to iteration to become the final system [10].

Each iteration goes through all aspects of software development, i.e., all process components, although with a different emphasis on each process component depending on the phase. This is depicted in the diagram in the beginning of section 'Process Overview'. The main consequence of this iterative approach is that the artifacts we described earlier grow and mature as time flows.

Process Components

The Objectory process is composed of 7 process components, which are described in terms of activities, workflows, workers and artifacts. There are four engineering process components:

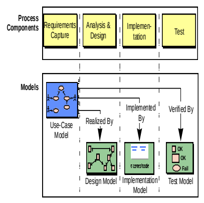
Requirement capture, Analysis and Design, Implementation and *Test*

and three supporting components:

Management, Deployment, and *Environment*

Process Components and Models

Each engineering process component describes how to create and maintain a model. Objectory has the following model: use-case model, design model, implementation model, and test model. The next figure shows the relationship of the process components and models.



Each process component is associated with a particular model.

Requirements Capture

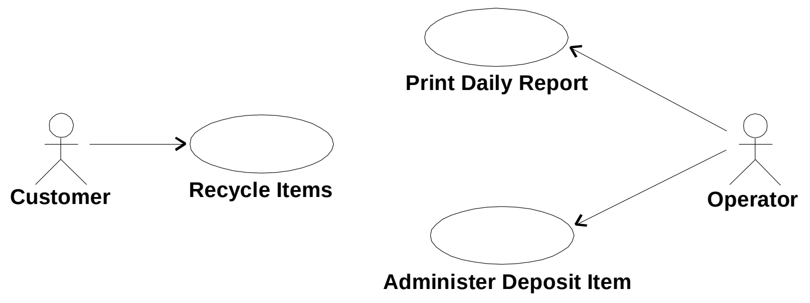
The goal of the Requirements Capture process component is to describe *what* the system should do and allows the developers and the customer to agree on that description. To achieve this, we delimit the system, define its surroundings and the behavior it is supposed to perform. Customers and potential users are important sources of information as well as any system requirements that may exist.

Requirements capture results in a *use-case model* and some supplementary requirements. The use-case model is essential for both the customer, who needs the model to validate that the system will become what he expected, and for the developers, who need the model to get a better understanding of the requirements on the system.

The use-case model is relevant to all people involved in the project.

The use-case model consists of *actors* and *use cases*. Actors represent the users, and any other system that may interact with the system being developed. Actors help delimit the system and give you a clearer picture of what it is supposed to do.

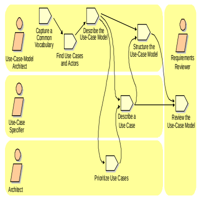
Use cases represent the behavior of the system. Because use cases are developed according to the actor's needs, the system is more likely to be relevant to the users. The following figure shows an example of a use-case model for a recycling-machine system.



An example of a use-case model with actors and use cases.

Each use case is described in detail. The *use-case description* shows how the system interacts step by step with the actors and what the system does.

The use cases function as a unifying thread throughout the system's development cycle. The same use-case model is used during requirements capture, analysis & design, and test.


Use-Case-Model Architect
Capture a Common Vocabulary
Find Use Cases and Actors
Describe the Use-Case Model
Structure the Use-Case Model
Requirements Reviewer
Use-Case Specifier
Describe a Use Case
Review the Use-Case Model
Architect
Prioritize Use Cases

The workflow in requirements capture, shown in terms of workers and their activities. The arrows indicate a logical order between the activities.

Analysis & Design

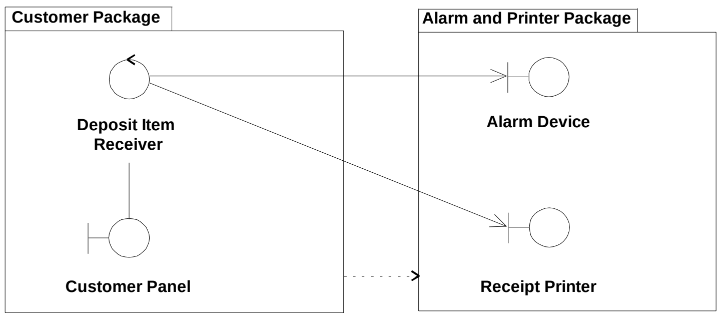
The goal of the Analysis & Design process component is to show *how* the system will be *realized* in the implementation phase. You want to build a system that:

- Performs—in a specific implementation environment—the tasks and functions specified in the use-case descriptions.
- Fulfills all its requirements.
- Is structured to be robust (easy to change if and when its functional requirements change).

The use-case model is the basis for design, along with the supplementary specifications.

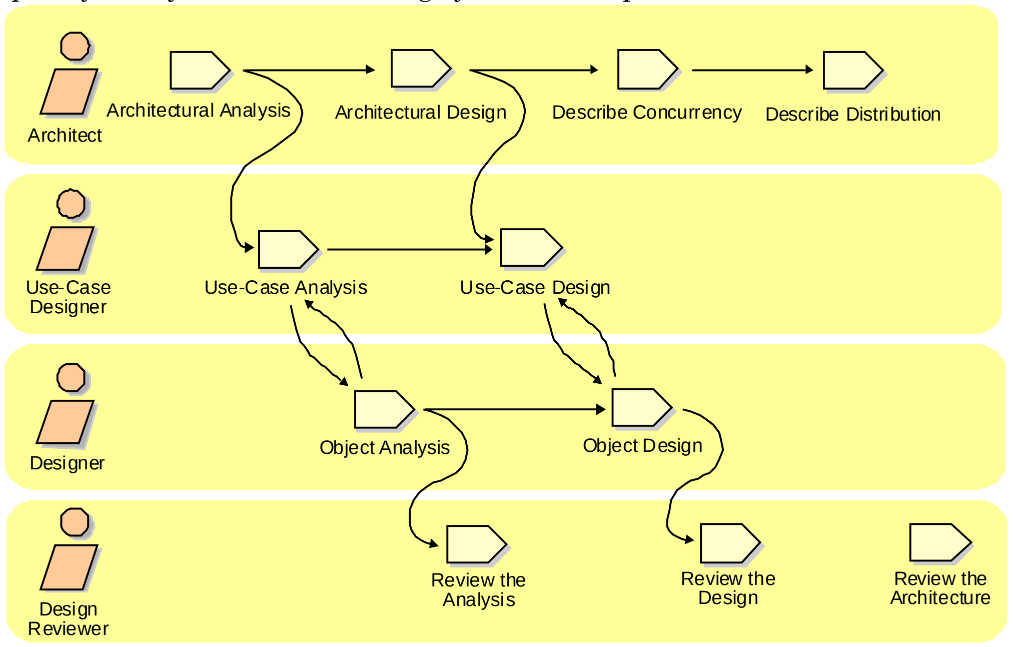
Analysis & Design results in a *design model* that serves as an abstraction of the source code; that is, the design model acts as a 'blueprint' of how the source code is structured and written. Design also results in 'inside-view' descriptions of the use cases, or use-case realizations, which describe how the use cases are realized in terms of the participating objects/classes.

The design model consists of design classes structured into design packages; it also contains descriptions of how objects of these design classes collaborate to perform use cases. The next figure shows part of a sample design model for the recycling-machine system in the use-case model shown in the previous figure.



Part of a design model with communicating design classes, and package group design classes.

The design activities are centered around the notion of *architecture*. The production and validation of this architecture is the main focus of early design iterations. Architecture is represented by a number of architectural views [9]. These views capture the major structural design decisions. In essence architectural views are abstractions or simplifications of the entire design, in which important characteristics are made more visible by leaving details aside. The architecture is an important vehicle not only for developing a good design model, but also for increasing the quality of any model built during system development.

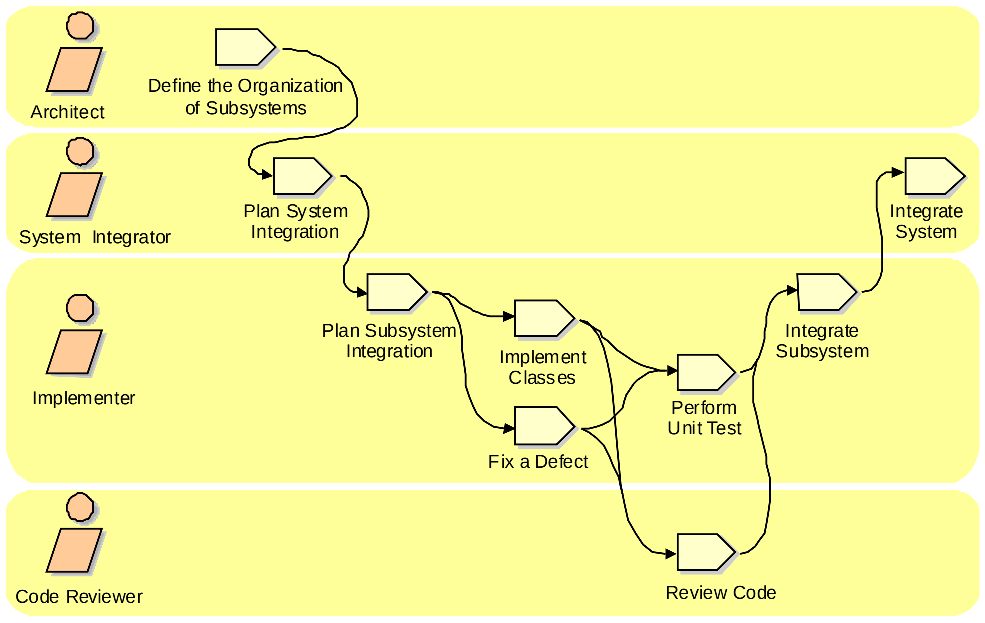


The workflow in analysis & design, described in terms of workers and their activities. The arrows indicate a logical flow between the activities.

Implementation

The system is realized through implementation producing the *sources* (source-code files, header files, makefiles, and so on) that will result in an executable system. The sources are described in an *implementation model* that consists of modules structured into implementation packages. The design model is the basis for implementation.

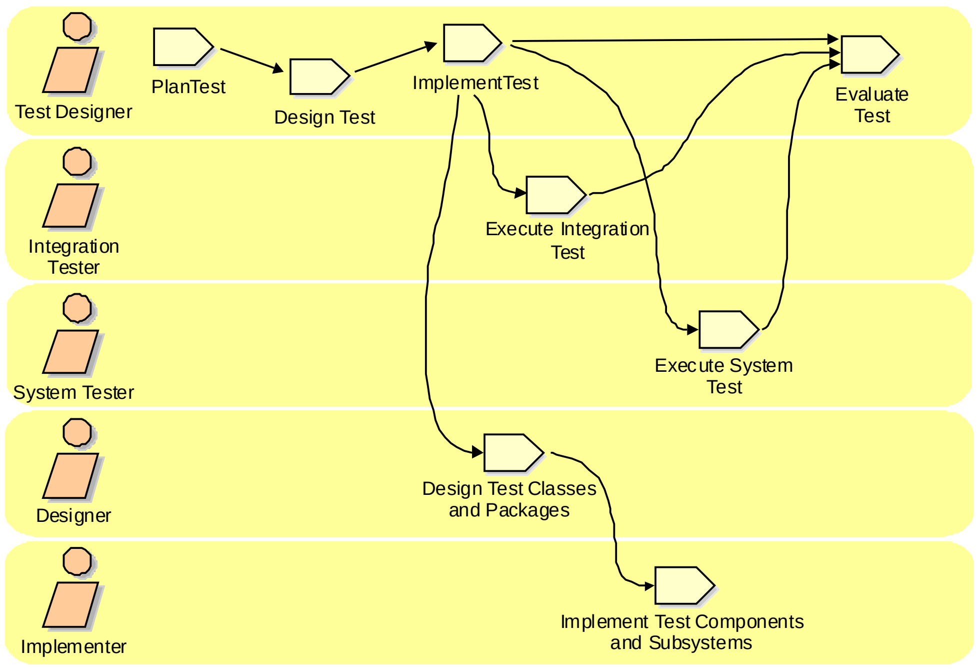
Implementation includes testing the separate classes and/or packages, but not testing that the packages/classes work together. That is described in the next process component, “Test”.



The workflow in implementation, shown in terms of workers and their activities. The arrows indicate a logical order between the activities.

Test

Test verifies the entire system. You first test each use case separately to verify that its participating classes work together correctly. Then you test (certain aspects of) the system as a whole with use-case descriptions as input to this test. At the end of test, the system can be delivered.



The workflow in test, shown in terms of workers and their activities. The arrows indicate a logical order between the activities.

Features of Objectory

This section explains the core ideas behind the Objectory process, its most salient features.

Object Technology

Many projects today employ object-oriented programming languages to obtain reusable, change-tolerant, and stable systems. To obtain these benefits, it is even more important to use object technology in design. Objectory produces an object-oriented design model that is the basis for implementation [3, 7, 8].

An object-oriented model aims at reflecting the world we experience in reality. Thus, the objects themselves often correspond to phenomena in the real world that the system is to handle. An object can be an invoice in a business system or an employee in a payroll system, for example.

A model correctly designed using object technology is

- *Easy to understand*. It clearly corresponds to reality.
- *Easy to modify*. Changes in a particular phenomenon concern only the object that represents that phenomenon.

Use-Case-Driven Development

It is often difficult to tell from a traditional object-oriented system model how a system does what it is supposed to do. We believe this difficulty stems from the lack of a "red thread" through the system when it performs certain tasks. In Objectory, *use cases* are that thread because they define the behavior performed by a system. Use cases are not part of "traditional" object orientation, but their importance has become more and more apparent. Other object-oriented methods provide use cases but use different names for them, scenarios, threads.

Objectory is a ìuse-case driven approach.î What we mean by that is the use cases defined for a system are the basis for the entire development process. Use cases play a role in each of the four engineering process components: requirements analysis, design, implementation, and test.

- The *use-case model* is a result of requirements analysis. In this early process we need the use cases to model what the system should do from the userís point of view. Thus, use cases constitute an important fundamental concept that must be acceptable to both the customer and the developers of the system.
- In design *use-case descriptions* are used to develop a design model. This model describes, in terms of design objects, the different parts of the implemented system and how the parts should interact to perform the use cases.
- During implementation the *design model* is the implementation specification. Because use cases are the basis for the design model, they are implemented in terms of design classes.
- During test the use cases constitute *test cases*. That is, the system is verified by

performing each use case.

Notice that a use case has several descriptions. For each use case there is a use case description, which describes what the system should do from the userís point of view, and there is a use case design, which describes how the use case is performed in terms of interacting objects.

Use cases have other roles as well:

- They can be used as a basis for iterative development.
- They form a foundation for what is described in user manuals.
- They may be used as ordering units. A customer can get a system configured with a particular mix of use cases, for example.

Controlled Iterative Development

The Objectory iterative approach is generally superior to a linear or waterfall approach for many reasons:

It lets you take into account changing requirements. The sad truth is that requirements *will* normally change. Requirements change and requirements "creep" have always been a primary source of project trouble, leading to late delivery, missed schedules, unsatisfied customers, and frustrated developers.

Integration is not one "big bang" at the end—elements are integrated progressively. Actually the Objectory iterative approach is almost continuous integration. What used to be a big, uncertain and painful time taking up to 40% of the total effort at the end of a project, is now broken down into 6 to 9 smaller integrations that begin with far fewer elements to integrate.

It lets you mitigate risks earlier because integration is generally the only time risks are discovered or addressed. As you unroll the early iteration you go through all process components, exercising many aspects of the project: tools, off-the-shelf software, people skills, and so on. Perceived risks will prove to not be risks, and new, unsuspected risks will show up.

It provides management with a way to do tactical changes to the product; for example, to compete with existing products. You can decide to release a product with reduced functionality earlier to counter a move by a competitor, or you can adopt another vendor for a given technology.

It facilitates reuse, since it is easier to identify common parts as they are partially designed or implemented, instead of identifying all commonality up front. Identifying and developing reusable parts is hard. Design reviews in early iterations allow architects to identify unsuspected potential reuse and develop and mature common code in subsequent iterations.

It results in a more robust architecture because you are correcting errors over several iterations. Flaws are detected even in the early iterations as the product moves beyond inception. Performance bottlenecks are discovered at a time when they can still be addressed, not on being discovered on the eve of delivery.

Developers can learn along the way, and the various competencies and specialties are more fully employed during the whole life cycle. Testers start testing early, technical writers write early, and so on. In non-iterative development the same people would be waiting around to begin their work, making plans and honing their skills. Training needs or the need for additional (perhaps external) help is spotted early on, during assessment reviews.

The process itself can be improved, refined along the way. The assessment at the end of an iteration not only look at the status of the project from a product/schedule perspective but also analyze what should be changed in the organization and in the process itself to perform better in the next iteration.

Project managers often resist the iterative approach, seeing it as a kind of endless ìhacking.î In Objectory, the interactive approach is very controlled; iterations are planned, in number, duration, and objective. The tasks and responsibilities of the participants are defined. Objective measures of progress are captured. Some rework does take place from one iteration to the next, but this, too, is carefully controlled.

Requirements Management

The two key elements behind a controlled iterative process are requirements management and change control. Requirements management is a systematic approach to eliciting, organizing, communicating and managing the changing requirements of a software intensive system or application.

The benefits of effective requirements management include:

- Better control of complex projects:

lack of understanding of the intended behavior as well as "requirements creep" are common factors in out-of-control projects.

- Improved software quality and customer satisfaction:

the fundamental measure of quality is "does this system do what it is supposed to do?" This can be assessed only when all stakeholders have a common understanding of what must be built and tested.

- Reduced project costs and delays:

error in requirements are very expensive to fix; decreasing these errors early in the development cycle cuts projects costs and schedule.

- Improved team communication:

requirements management facilitates early involvement of users to ensure that the application meets their need; well managed requirements builds a common understanding of the project needs and commitments among all stakeholders: users, customers, management, designers, testers.

Focused more closely towards the needs of the development organization, change control is a systematic approach to managing changes in requirements, design, implementation, but also covers the important activities of keeping track of defects, misunderstandings, project commitments, and being able to associate these with specific artifacts and releases.

A Strong Emphasis on Architecture

Use cases drive the Objectory process end-to-end over the whole lifecycle, but the design activities are centered around the notion of *architecture*—system architecture, or for software-intensive systems, software architecture. The main focus of the early iterations of the process—mostly in the elaboration phase—is to produce and validate a *software architecture,* which in the initial development cycle takes the form of an executable architectural prototype that gradually evolves to become the final system in later iterations [9].

The Objectory process provides a methodical, systematic way to design, develop and validate an architecture. It offers templates for architectural description around the concepts of multiple architectural views, and the capture of architectural style, design rules, and constraints. The design process component contains specific activities aimed at identifying architectural constraints and, architecturally-significant elements, as well as guidelines on how to make architectural choices. The management process shows how the planning of the early iterations takes into account the design of an architecture and the resolution of the major technical risks.

Architecture is important for several reasons:

It lets you gain and retain intellectual control over the project, to manage its complexity, and to maintain system integrity.

A complex system is more than the sum of its parts, more that a succession of small independent tactical decisions. It must have some unifying coherent structure to organize those parts systematically, and provide precise rules on how to grow the system without having its complexity explode beyond human understanding.

The architecture establishes the means for improved communication and understanding throughout the project by establishing a common set of references, a common vocabulary with which to discuss design issues.

It is an effective basis for large-scale reuse.

By clearly articulating the major components and the critical interfaces between them, an architecture lets you reason about reuse, both internal reuse—the identification of common parts—and external reuse—the incorporation of ready made, off-the-shelf components. But it also allows reuse on a larger scale: the reuse of the architecture itself in the context of a line of products that addresses different functionality in a common domain.

It provides a basis for project management.

Planning and staffing are organized along the lines of major components. Fundamental structural decisions are taken by a small, cohesive architecture team; they are not distributed. Development is partitioned across a set of small teams each responsible for one or several parts of the system.

Component-Based Development

A software component can be defined as a non trivial piece of software, a module, a package or a subsystem, that fulfills a clear function, has a clear boundary and can be integrated in a well-defined architecture. It is the physical realization of an abstraction in your design.

Components come from different avenues:

- In defining a very modular architecture, you identify, isolate, design, develop and test well-formed components. These components can be individually tested and gradually integrated to form the whole system.
- Furthermore, some of these components can be developed to be reusable, especially the components that provides common solutions to a wide range of common problems. These reusable components which may be larger than just collections of utilities or class libraries, form the basis of reuse within an organization, increasing overall software

productivity and quality. [8]

- More recently the advent of commercially successful component infrastructures such as CORBA, the Internet, ActiveX or JavaBeans, triggers a whole industry of off-the-shelf components for various domains, allowing to buy and integrate components rather than developing them in-house.

The first point exploits the old concepts of modularity, encapsulation, bringing the concepts underlying object-oriented technology a set further. The last two points shift software development from programming software (a line at time) to composing software (by assembling components).

Objectory supports component-based development in several ways.

- The iterative approach allows to progressively identify components, decide which one to develop, which one to reuse, which one to buy.
- The focus on software architecture allows to articulate the structure: the components and the ways they integrate: the fundamental mechanisms and patterns by which they interact.
- Concepts such as packages, subsystems, layers are used during analysis and design to organize components, specify interfaces.
- Testing is organized around components first, then gradually larger set of integrated components.

Process Configurability

The *Rational Objectory Process* is general and complete enough to be used “as is” by some software development organizations. However in many circumstances, this software engineering process will need to be modified, adjusted, tailored to accommodate the specific characteristics, constraints and history of the adopting organization. In particular a process should not be followed blindly, generating useless work, producing artifacts that are of little added value; it must be made as lean as possible and still be able to fulfill its mission to produce rapidly and predictably high quality software.

The process elements that are likely to be modified, customized, added or suppressed include: artifacts, activities, workflows, workers and process components.

The Origin of Objectory

Objectory has many different sources. The more essential of these are:

- Objectory was originally developed in Sweden by Dr. Ivar Jacobson at Objectory AB. Centered around the concept of *use case* and *object-oriented design* method, it has gained recognition in the software industry and has been adopted and integrated by many companies world-wide. A simplified version was published as a book in 1992 [7].
- The Rational Approach is an *iterative* process, focused on *software architecture*. It has been developed by several different people at Rational, including Philippe Kruchten, Grady Booch and Walker Royce. Various papers [9], [10] and books [3] has described this approach. The Rational Approach was integrated with Objectory

in 1996.

- SQA Process is a formal test methodology developed by SQA, acquired by Rational SW in early 1997. It is a leading test methodology for the Windows platform. SQA Process was integrated with Objectory in 1997.
- Requirements College is a leading requirement management methodology developed by Dean Leffingwell et.al. at Requisite. Requisite was acquired by Rational early 1997. Requirements College was integrated with Objectory in 1997.

The Objectory Process is supported by Rational's leading methodologists, including Grady Booch, Ivar Jacobson, Philippe Kruchten, Dean Leffingwell, Walker Royce and Jim Rumbaugh.

Objectory is sold as a product by Rational Software Corp. It is available in on-line, browsable form and in printed book from. It is supported by training courses and other services.

References

1. Barry W. Boehm, “A Spiral Model of Software Development and Enhancement,” *Computer,* May 1988, IEEE, pp.61-72

2. Barry W. Boehm, “Anchoring the Software Process,” *IEEE Software,* 13, 4, July 1996, pp. 73-82.

3. Grady Booch, *Object Solutions,* Addison-Wesley, 1995.

4. Grady Booch, Ivar Jacobson, and James Rumbaugh, *Unified Modeling Language*, White paper, Rational Software Corp., 1996.

5. Alan W. Brown (ed.), *Component-Based Software Engineering*, IEEE Computer Society, Los Alamitos,CA, 1996, pp.140.

6. Michael T. Devlin, and Walker E. Royce, *Improving Software Economics in the Aerospace and Defense Industry,* Technical paper TP-46, Santa Clara, CA, Rational Software Corp., 1995

7. Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Övergaard, *Object-Oriented Software Engineering—A Use Case Driven Approach,* Wokingham, England, Addison-Wesley, 1992, 582p.

8. Ivar Jacobson, M. Griss, and P. Jonsson, *Software Reuse—Architecture, Process and Organization for Business Success,* ACM Press, New York, NY, 1997.

9. Philippe Kruchten, “The 4+1 View Model of Architecture,” *IEEE Software,* 12 (6), November 1995, IEEE, pp.42-50.

10. Philippe Kruchten & Walker Royce, “A Rational Development Process,” *CrossTalk*, 9 (7), STSC, Hill AFB, UT, pp.11-16.

Cf. also http://www.rational.com/products/objectory/process/

O.O.S.I. OBJECT ORIENTED SYSTEM INTEGRATION

PROJECT N. 10987

CARICCHIA PAOLO
AEROPORTI di ROMA

Description of the Company

Aeroporti di Roma is a company of the IRI group, born in 1974 from the fusion of several companies (ground services) acting in Fiumicino and Ciampino airports of Rome. Aeroporti di Roma (AdR hereinafter) is responsible for the co-ordination and rationalisation of the facilities management and ground services in the airports of Rome. In the next future AdR will become a private company, and this process has begun six month ago putting on the market the 45 % of the stocks. An important legal event pushed AdR management to renew their information system: that is, the Italian anti-trust committee forced the company to share the market with new handling agents. As a consequence, the AdR Information System has become a critical factor of success to compete on the market. In order to offer competitive services, AdR wants to improve the overall efficiency of their own organisation. In this respect, it is clear that IS plays a main role: its efficiency has an impact on the global level of service offered by the company.

AdR's IS department is organised as follow:

it includes132 AdR people plus external contractors.

The fields of activity are :

- SW development and maintenance.
- Technical infrastructures and automated system development.
- System management and HW maintenance
- Client Managers

The company policy regarding the software development is to use external labour for the low level activities or for "turn key" project.

Actually, we are developing 20 main software projects and about one hundred maintenance and implementation activities.

External contractors provide about 70 % of the total development needs.

Principal internal skill are :

Project manager
analyst
analyst programmer

The competence level and qualification are in the average.

The average is high, in some cases depending on the fact of a re-qualification of low level skilled personnel to different activities.

Project manages lead workteam of internal and external resources.

"Method and Resources" plans and controls the resources scheduling and supporting the workteam on methodology and tools.

In addition, it maintain Company data dictionary.

RAD is an organisation that develops small applications and manages all the items regarding P.C., LAN, and Internet activities.

Information system department support different business areas like:

1. airport handling
2. flight information
3. maintenance of buildings and plants
4. marketing activities
5. accounting and financial areas
6. infrastructures development.

Main problems are :

- integration among the different applications is very difficult and expensive;
- training costs are high, mostly dependent on the low profile of end-user.
- the heterogeneous user interface constitutes notable problems from the functional point of view.

Starting scenario

The main part of applications is mainframe-based, mostly developed in the last decade or before, for each different business area.

Usually, the user interface is 3270-like and databases are relational.

For admin. business area applications, database are hierarchical and data integrity is guaranteed by batch procedures.

Many applications are PC based, using data (via file transfer from mainframe databases) for personal elaboration and statistical analysis using office tools like those Windows based.

The mini-based applications are significantly used just for process-control applications, for CAD, SAP or document management system.

This leads to a great variety of different product and devices to be managed.

AdR has conducted the Object Oriented System Integration (OOSI hereinafter) application experiment with the support of the European Commission within the ESSI Project number 10987.

The OOSI objectives were to evaluate the effectiveness of the OO paradigm in terms of system interoperability, maintenance and evolution, through the development of a significant application, and to disseminate this culture within the organisation. In fact, the OMT methodology and the O2 system were used to develop the AdR

"**Weight and Balance**" system.

WORKPLAN

The experimentation was arranged according to the following path:

1. Assessment of the current information system.
2. Identification of a significant AdR pilot application, in terms of data complexity and distribution, to be re-designed and developed with the Object Oriented paradigm.
3. Development of the identified pilot application.
4. Dissemination of the Object Oriented paradigm within the organisation.
5. Assessment of the experimentation results with respect to the OOSI objectives.

The preceding activities were structured into the following work packages :

WP1 - Object Oriented analysis of subset of existing information system (1, 2)
WP2 - Object Oriented design of the integration platform (3)
WP3 - Evaluation and Dissemination (4, 5)
WP4 - Project Management

The new "**Weight and Balance**" system is composed by five subsystems (the acronyms derive from Italian language translation of the description in brackets):

GEMO	(Operative Manuals Manager)
GTT	(Technical Tables Manager)
SVPL	(Work Balancing Manager)
BIL	(Load Balancing Manager)
GMID	(Mainframe Communication Manager)

The functionality of each subsystem will be described during the presentation.

EXPECTED OUTCOMES

The main expected results from OOSI were:

An evaluation of the maturity and effectiveness of the Object Oriented paradigm in terms of application level interoperability, maintainability and evolution.
The dissemination of the Object Oriented paradigm within AdR

Application level interoperability, intended as the design of a comprehensive entities conceptual schema of the AdR Information System, was considered a first fundamental step towards technology level interoperability, that is the possibility to access data in the heterogeneous databases on different hardware and software platform. This latter was beyond the scope of OOSI.

The use of the Object Oriented paradigm to develop a general entities conceptual schema for AdR information system was expected to facilitate the evolution of the existing system and the development of the new ones.

THE IMPLEMENTATION OF THE IMPROVEMENT ACTIONS

The OOSI experiment was an important opportunity for AdR to learn through direct experience about client-server architecture, graphical user interfaces, the object oriented paradigm, and the beneficial effects of these technologies from the organisational perspective as well. In particular, the object oriented paradigm has shown itself to be powerful and very promising in respect of the possibility to provide a comprehensive, uniform and structured view of the AdR application domain and to evolve this view in a very easy way. The object oriented database schema developed in the project has been evolving very quickly in a period of few month. AdR are now more confident that re-engineering their information system and developing the new applications on client server environment, equipped with the most advanced software technology can improve their services and turn into competitive advances in the medium-large term.
AdR are now dealing with education of the technical staff and revision of the software process organisation. Moreover, IS department is now in change of careful looking at emerging standards on object oriented methodology and technology side. In particular, AdR are now paying attention to the emerging *Unified Modelling Language* and the supporting technology, which will probably be a subject of a new experimentation in the near future.

THE MEASURED RESULTS AND THE LESSON LEARNED

The application experiment was based on a real and significant AdR application. From AdR perspective, this is a crucial point and reinforces their confidence on the validity of the experiment results obtained from OOSI. The strength of OOSI also stems from the fact that the experiment has not been confined to a limited group of specialist, but various departments has been involved in it. Finally from the OOSI experiment, the following lessons can be drawn:

- The rapidity and simplicity with which new domain entities (O2 classes) were added and tested to refine the Weight and Balance system database schema, by using specialisation, aggregation, and classification abstraction mechanisms, leads to the consideration that application maintenance and evolution are facilitated using object oriented technology.
- Object oriented paradigm allows for major data integration. Even end-user technical manuals could be easily stored in the application database and associated with the concerned entities.
- Object oriented modelling and tools make possible to develop system starting from a high level view of application domain, which permit to overcome to the language gap existing between end-users and software designer.

This project allowed us to experiment with success :

- A new organisational model
- A client-server HW/SW architecture

- A modern and powerful software methodology and technology

However, careful attention must be paid to the introduction of these new technological and organisational paradigms within the company. The process must be gradual and involve as many people as possible, in order to maximise the consensus and reduce resistance from people less ready to accept any change.

Session 7 – SPI Experience for Small Teams

A Small Software Developers' framework for evaluating the internal usage of IT

Béatrix BARAFORT, Anne HENDRICK,
Philippe LIEMANS, Jean-Pol MICHEL

Joint contribution from the Software Engineering team

Centre de Recherche Public Henri Tudor, L-1359 Luxembourg

CMM in a Micro Team: A Case Study

Joao Batista

ISCAA/CISUC, Portugal

A. Dias de Figueiredo

CISUC, Portugal

Improving Estimation and Requirements Management: Experiences from a very small Norwegian Enterprise

Svein Are Martinsen (sveinare@invenia.no)

Invenia AS, P.O. Box 282, N-9201 Bardufoss, Norway

Arne-Kristian Groven (groven@nr.no)

Norwegian Computing Center (NR), P.O.Box 114 Blindern,

N-0314 Oslo, Norway

A Small Software Developers' framework for evaluating the internal usage of IT

Béatrix BARAFORT, Anne HENDRICK,
Philippe LIEMANS, Jean-Pol MICHEL
Joint contribution from the Software Engineering team
Centre de Recherche Public Henri Tudor, L-1359 Luxembourg

Introduction

Managing efficiently the Information and Communication New Technologies is a prerequisite in order to support the access of small companies in the Global Information Society.

The Software Market for the French speaking region composed of the Grand Duchy of Luxembourg, provinces of Luxembourg, Namur and Hainaut in Belgium, and the French Lorraine, could be described schematically as two highly independent layers. The first layer, “High Level Software Market”, links big firms and organisations (European Commission, banks and insurance, industry) with computer manufacturers, software houses and consulting firms (either medium size independent companies or subsidiaries of international groups). The second layer, “Low Level Software Market", connects Small and Medium sized Enterprises(SME)’s, with or without Information Technology (IT) department, with small local software house. The latter is potentially larger but presents several weaknesses: very little formalised Customer/Supplier relations, lack of visibility, few consulting services, high risks in terms of reliability and continuity.

These reasons conducted the Software Engineering team of the Centre de Recherche Public Henri Tudor (public research centre devoted to innovation and technology transfer) in Luxembourg to define the SPIRAL*NET project 0. Its objectives are to optimise and to generalise best practices related to the Customer/Supplier processes and the associated support ones.

This paper describes experiences and case studies collected from various IT projects. Based on conclusions and lessons learned, it outlines what could be a framework to evaluate the maturity of Small Software Developers (SSD) companies in the management of internal usage of IT.

Panorama of quality dedicated approaches

All along the 20th century and particularly in its second half, quality approaches contributed to the development of companies : TQM initiatives, ISO 9000 certifications 000, or competing for quality awards. For the last decade, IT-dedicated quality models appeared, more specifically based on the process concept (CMM : Capability Maturity Model, Bootstrap, Trillium, ISO 12207, ISO 15504 0 also known as SPICE : Software Process Improvement and Capability dEtermination).

The majority of the previously mentioned standards are stemming from military, industrial and aerospace initiatives, supported by governments and main industries. These standards are directly applicable to big firms, but with lot's of difficulties to SSD's and SME's, principally for cost reasons. Facing the growing importance of IT in the enterprises and the competitive context of the end of the century, software process improvement has become a key success factor, whatever the companies size is. More and more adaptations are proposed for performing assessments and improving software processes in small IT structures. Relating to excellence models, the EFQM (European Foundation for Quality Management) developed a model for SME's (The European Model for Small and Medium sized Enterprises), with an associated award (the European Quality Award for SME's). Concerning the CMM, a company named LOGOS has worked out a lightened approach 0. The European Commission is bringing out programs which aim at developing such approaches in SME's (for example projects such as SCATE 0, SPIRE 0, TAPISTRY 0 0, BIG Project in the ESI -European Software Institute- 0). In Wallonie (French speaking part of Belgium), several initiatives are performed in order to assist SME's and to provide them with tools and methods tailored to their context 0.

The regional context

The region which concerns us constitutes a geographical, cultural and economic uniform unit. It is characterised by the use of a common language, French, and by a geographical and historical proximity. Concerning economy, surveys conducted by CEPS-INSTEAD (Centre d'Etudes de Populations, de Pauvreté et de Politiques Socio-Economiques / International Networks for Studies in Technology, environment, Alternatives, Development showed that the area has been seriously affected by industrial decline, and the iron and steel industry recession had cost it dear. An important tertiary sector is now growing up. This region is characterised by economical poles of development around main cities (Nancy, Luxembourg, Namur and Charleroi), and by an important network of SME's.

The following paragraph illustrates the awareness of local SME's to Software Process Assessment (SPA) and Software Process Improvement (SPI). The study realised by CITA 0 gives similar indications on Belgian French speaking SMEs.

Description of the panel

During several missions between June 97 and August 98, the CRP Henri Tudor teams have met a valuable number of luxemburgish SSD's. The following data concerns about 16 of them.

The initial observation allows a classification of these SSD's in 3 categories :

the IT professionals (5 of them) : these are mainly local companies and are specialised either in software development (2), or propose full services from software development to LAN / WAN infrastructure (3).

the companies with a well set IT department (both organisational and people aspects) (5 of them) : they are all internationally aimed ; 4 are partners or subsidiaries of a larger group. They have a well set IT strategy with home software developments (3) or implementation of an ERP (2). The computer team works by its own, but can be closely supported for the largest projects by the main company of the group (2) ; this

makes the team size variable, but usually it doesn't reach 20 people.
the SME's where the IT department is managed by a « one-man-band » helped by a maximum of 3 collaborators (6 of them). Most of them outsource the software developments and a large part of the activities of computer support.

This panel of companies consists of 10 industrial companies (polymers, machinery pieces, materials, high added value steels, etc.) and 6 in the services field.

For 7 companies, the technological environment can be qualified of well developed and efficient (well designed LAN, E-mail and intranet functions). On the other hand, 3 companies have significant gaps like an uncomplete LAN, no inter site links and/or important lacks in the available software applications. In each case, the will or plans exist to solve these problems. For 6 companies, the computerisation maturity is medium with strengths but also with at least one large weakness such as a neglected IT department, the computerised operational works not automated, the main software unchanged for years and largely obsolete, etc.

States of commitment for process improvement

All companies want to improve and have the consciousness of making it an ongoing process; the differences are the amount of energy they put in it and the speed of the implementation of the selected changes.

The last remarkable point is that nearly half of the companies hide their face when the time is up to fight the shortcomings and problems : taboos, ideas which never become reality, resistance of several kinds, other priorities… All theses thoughts emphasise the complexity of the problems, and highlight the global lack of an integrated IT strategy in the companies, and particularly in SSDs.

SPA and SPI case studies

The Centre Henri Tudor has chosen the ISO 15504 framework to support assessments and SPI activities. Located at the tactical level in an organisation, process assessment bridges the gap between operational and daily tasks formalised by procedures from a quality system (i.e. ISO 9001/9002 compliant), and the strategic view of the company featured by business excellence models such as EFQM, Baldrige…

In order to give prominence to this tactical turning point, the assessment preparation phase consists in determining value chain activities of the organisation in connection with its vision and its mission. Then, for each activity, key success factors induce the selection of critical processes for the organisation. These processes can be assessed and used as a basis to build a software practices improvement program.

Coming from the ISO 15504 baseline, an ISO 15504 compliant process model and the associated improvement approach 0 are adapted to our local context. The following paragraph illustrates these remarks via a description of selected case studies. An ISO 15504 assessment has been realised for the company described in each of them. Today, two of these case studies have been implemented in an improvement plan.

Assessments automatically happened according to the following steps (the case studies will only stress the outstanding elements of each step) :

initiation : introduction of the firm, awareness on SPI, selection of key-processes preparation : determination of the field and the sample of activities to assess according to the selected processes, planning of the interviews

assessment interviews (lead by a pair of assessors) and consolidation of raw results, progressive definition of the assessment profile

analysis and reporting : results analysis, reports writing : detailed synthesis for the managers

results presentation : to the managers and the concerned staff

assessment : validation of the method and results, proposition of a SPI program to the firm

As for the advice in the definition and implementation of a SPI plan, the CRP Henri Tudor operated in two different ways : either as a major actor of assistance to a firm (for specific actions of assistance and advice during the preparation and the implementation of a SPI program), or by participating to the SCATE 0 project as a logistical relay and as an observer during the training sessions. The SCATE program consisted in a series of nine training/action sessions during nine months destined to SSD. This program was meant to train one person (*the champion*) to the suitable techniques in SPI, to enable firms to acquire the competencies, abilities and behaviours required to implement the change within the firm, and eventually to give a chance to share the experience with other firms which are involved in the same process. The program was based on the CMM and particularly on the level 2 Key Process Areas. Two working groups gathering 5 firms each took place in Namur (B) and in Luxembourg.

Case study A

Firm context – SPI context

Firm A is a Belgian SME which employs 250 persons. It gathers 3 non-profit-making associations, which are the Social Insurance Fund, the Social Secretariat, and the Family Allowance Fund.
The IT department is a common unit for the 3 non-profit-making associations. It gathers 21 persons and is divided into 3 functional teams, according to the non-profit-making associations. The technological environment is centralised (mainframe system).
The manager of firm A attended a symposium about a program of training/action in the SPI field for the SME's. He decided to involve its company in such a program and initialised an improvement project (participation to a SCATE User Group of Luxembourg and Namur). In order to know the state of the software practices in his firm before starting the program, an ISO 15504 assessment was made.

Outcome for the ISO 15504 assessment

<u>Weaknesses:</u>
Expression of the vision / mission / value chain and selection of the key processes (not enough time spent on this step; no distance)
Embarrassing presence of an observer from the IT management for the interviewed users.
<u>Strengths:</u>
Total support from the sponsor and implication of all Project Managers and of two key-users
Building of a process which is adapted to the firm's context and to the interviewed persons (Requirements and Tests process destined to users)
Results presentations used as a way of making every one aware of the role of IT within the firm and underlining of the efforts made to improve software practices.
As for CRP Henri Tudor, experience and tools acquired from former assessments.
<u>Lessons learned:</u>
Need to improve the step of selection of key-processes in the firm (provide an assistance to the formulation of the vision/mission/value chain) and to drive the thought towards other processes than engineering ones.
Proposal of an assistance to the implementation or the evolution of the IT strategy in the firm.
Need to precise the recommendations and to start the preparation of an improvement plan.
Making of a formal outcome of the assessment.

Outcome from the tutoring program experience

<u>Weaknesses:</u>
Terms of SPI difficult to understand
Confusions between SPICE and CMM
Lack of an external follow up for the planned improvement actions
<u>Strengths:</u>
Awareness of the usefulness of SPI and motivation of the staff already acquired during the ISO 15504 assessment
Formalisation (for the planning and the follow up) and Change Management (especially for the less young persons of the team)
Sharing of diverse experiences with other firms during monthly training sessions
Desire from the champions (supported by the sponsor) to continue the program of improvement beyond the SCATE Program
<u>Lessons learned:</u>
Need of a structure for improvement and of a precise terminological frame which would be already applied before starting the Program.
Need of external assistance to valid the implementation of improvement actions
Need of an external catalyst to avoid slowness of the step and decline of enthusiasm
Need to talk about the functions of technology watch with the firm (these functions could be common to several companies and SSD)

General conclusion

The improvement program engaged via the SCATE tutoring project was not ended during the last training sessions. Nevertheless, the managers and the champions decided to continue their work and to enlarge their actions to other IT activities, such as the Requirements Management and the Quality Assurance (already stressed during the ISO 15504 assessment). It is possible that a new ISO 15504 assessment will be done later, in order to measure precisely the evolution of software practices.

Case study B
Firm context - SPI context
Firm B is a SME of about 170 persons. It is a small business bank in Luxembourg. A particular and independent organisational unit makes the IT treatments of the firm. This unit plays the role of a privileged software house for B but it has also developed external services such as facilities management which ISO 9002 certified. The IT staff gathers 24 persons. The activities of the IT team deal with development and help-desk service. The technological environment is centralised (mainframe system). Following the SPIRAL'97 conferences in Luxembourg, a member of the Management Committee of this company has been interested in the idea of an ISO 15504 assessment made by a neutral and independent institute, such as the CRP Henri Tudor. Once the management convinced and particularly the IT manager (sponsor of the service), an assessment of the software practices has been decided with the following goals : determination of the maturity level of the development process, identification of strengths and weaknesses, risks and improvement opportunities, recommendations of precise measures which will be useful to the working-out of an improvement program, and of the staff awareness to the continuing improvement of software practices.
Outcome for the ISO 15504 assessment
<u>Weaknesses:</u> Expression of the vision / mission / value chain and selection of the key processes (not enough involvement from the firm in the step of selection of the processes and not enough follow up from the CRP Henri Tudor) Not enough support from the sponsor and not enough implication to determine the possible continuation of the assessment with the definition of a whole improvement project Questions linked to the IT strategy asked by the Management Committee Complex relations between the organisational unit which plays the role of a software house for the bank and the users themselves (difference of goals and no contract between client and supplier). <u>Strengths:</u> Involvement of all Project Managers and of three users from the bank. Building of a tailored process to the context of the firm and to the interviewed persons (IT engineers and end-users group process) As for CRP Henri Tudor, experience and tools acquired from former assessments. Proposition of the outline of a action plan based on the recommendations of improvement. <u>Lessons learned:</u> Review planning ratio to have a good understanding of the business activities Extend preparation time for the assessment in the case of building processes for the assessment requirements. Need to improve the step of selection of key-processes in the firm (provide an assistance to the formulation of the vision/mission/value chain) and to drive the thought towards other processes than engineering ones. Advice to the firm all the way through. Proposal of an assistance to the implementation or the evolution of the IT strategy in the firm.
General conclusion
As a whole, the assessment has been welcomed by the involved team (the IT staff) and happened in good conditions. Nevertheless, the sponsor has not fully played his role, therefore it has been difficult to prove him the need to define and implement a SPI project. Four months after the assessment, the improvement project has still not started.

Case study C
Firm context - SPI context
Firm C is part of a European group (about 220 persons in Europe) which works in the manufacture of colours concentrates and in colourings for plastics. There is also a group in the United States. As a whole, the company knows a great growth. The IT staff of the European subsidiary (for the components of Luxembourg and Belgium) has recently grown to 6 persons (including 4 developers). The technological environment is UNIX. Following an awareness event for the SCATE program, the manager of firm C has decided to involve it in such a program (SCATE User Group of Luxembourg). Indeed, this firm has a few means to reach the same stakes as the great companies. IT engineers work as heroes with no view on the development.
Outcome for the ISO 15504 assessment
Weaknesses: Terms of SPI difficult to understand Difficult assimilation of the CMM guide (4 months were needed) Strengths: The presentation of the ami 0 (application of metrics in industry) method endorsed the internal step (generalised to the whole company) of implementation of metrics (issued from a Goal/Question/Metric approach). After the time of assimilation of the CMM guide 0, practical elements were drawn out applicable to the context of the firm (in terms of involvement, of common and systematic step) Lessons learned: Need of a toolbox (not enough tools provided via SCATE) Need of an external assistance to the firm (during and after SCATE)
General conclusion
After the end of the training sessions, the *champion* and the managers were decided to continue the improvement program (not finished at the end of the sessions). After the jolt of the past, the future goals are to manage the flow of software processes used by the IT team and to have the view on these processes. The goal is settled on year 2000 modifications. This company made the deliberate choice to lead small improvement actions, at high added-value (Quick Wins), which imply a long term partnership in the issue of projects/actions/training, the exchange of experiences and the assistance of improvement actions by an external organisation.

Specifications for a SSD's framework

Case Studies Results : contribution to an assessment and improvement approach

Several outstanding and recurrent facts arose in the case studies. Based on these experiences, some general guidelines and specific tips for SSD (our recommendations) can be proposed in order to lead the assessment and improvement actions, and to contribute to a global SSD's framework.

Awareness

It's of no use to perform an assessment if the awareness level and the approach support in the SSD is not sufficient :

Need for a preliminary awareness or training action for the sponsor, and at the very least, all staff involved in the assessment

Our recommendations :

To assist the assessment with a real awareness effort for each contacted person

To plan specific awareness action to the sponsor and assessment co-ordinator

To organise awareness action for all staff

To cover all projects in the SSD

To interview each person for at least one process

To organise assessment interviews with a group of persons who are not performing direct assessed process related tasks, but are customer or supplier for them.

Processes to assess

Generally we use the value chain diagram to identify the most critical core processes and IT key processes.

Important links between the IT department's vision and missions and the whole company's vision

Key processes choice with IT managers often ends up with engineering and project management processes

Our recommendations :

We think that a set of processes, even if they are partially implemented, have to be started on in a systematic way. They contribute to the value chain activities. We suggest a list of generic activities which are often present in SSD and applicable to the software context :

Product specification

Product selection and/or Product development

Product maintenance

Technology management

Management of activities

Considering the process model 0, the following list of processes which contribute to the core activities of most SSD, has been established (most of the processes are parent processes in the model) : Acquisition, Supply, Project Management, Risk Management, Change Management, Problem Management, Configuration Management, Infrastructure, Strategy Management, Development, Maintenance

We identified a particular process which does not exist in the model but which is essential considering the SSD identified needs : Elaboration or evolution of the IT strategy. This new business process is being established in CRP Henri Tudor and will be the baseline for assisting SSD in defining or modifying their IT strategy.

Quick Wins

Quick wins are improvement actions which don't require important implementation efforts, but with visible and rapid results in the firm. It can be a document template at disposal, a punctual involvement of a person in a meeting where he/she was never invited before, available information never consulted before.

Our recommendations :

To implement Quick wins which bring a lot of added value to IT activities, particularly for SPI aware companies with limited resources

Improvement actions

Well-known tips to manage SPI projects are critical in the context of SSD :

Need for setting an improvement infrastructure

Define an organisational frame for projects

Focus on key process related actions such as best practices collected by the means of working groups, improvement implementations in a pilot context, action refinement, institutionalisation and process formalisation (process definition)

Regular re-assessments

Our recommendations :

To identify an improvement infrastructure and to promote it to all staff in the SSD (Two main roles are identified : the sponsor and the SPI actor. The SPI actor may be project leader, process owner, and expert once at a time).

To start on systematically a set of processes, even if they are only partially implemented, such as : Elaboration or evolution of the IT strategy, Client/supplier relationships, problem management, change management, infrastructure

To assist the SSD to implement improvement actions (performed by an organisation such as CRP Henri Tudor). This seems essential in order to encourage similar experience reuse and a technology transfer adapted to their context

Guidelines for a framework

Based on experiences and case studies collected from different IT projects and various activity sectors, requirements for a framework dedicated to SSD are defined. This framework can be used **to assess how a company uses IT** and **to determine how new technologies could support its strategy and the development of its activities**.

First the framework is composed of the core processes identified through the analysis of the vision and the business goals of the SSD. The well known value chain diagrams helps you to model each adding value activity to the services and/or goods production cycle. Some supporting processes could also be identified. The drivers are sometimes called key success factors or aims; they outline the desired outcomes to successfully achieve the mission.

A second dimension of the framework consists in modelling the usage of IT via the identification of "information and communication processes". These are features for stored, transformed or exchanged information among core processes in order to better analyse the usage of Information and Communication Technologies and to appraise in what extent they efficiently support core processes related activities. Each of the information and communication process could be described through a 3-layer architecture to cover technical, software and organisational components. The technical components address hardware and networking (types of computers, networks, cabling, and network liaison). The software components covers fundamental applications (production, stocks, invoicing, accounting, salaries, human resource management) and networking integration (networking protocol, operating system,…). The organisation addresses the availability of computer resources, facilities management or outsourcing choices, selection of suppliers.

Nowadays the IT evolution in firms can be stated by 5 major IT application categories depending on the finality of the associated information system :

computerisation of operational activities
integration of subsystems
decision support systems
co-operation within the company or inter-personal communication
external exchanges or inter-professional communication

The third dimension concerns the key processes associated to IS projects (an ISO 15504 compliant process model 0 can then be used) where the processes are the baseline for assessment and improvement actions.

Figure BB&AH.1 below shows how the framework is articulated, with the current, target and potential IT view in a company.

During interviews with managers and employees, all data, templates, user guides are collected (all materials allowing a good knowledge of the core processes and the information and communication processes). Each of them is documented through a set of attributes and models.

All these data will be analysed through three different viewpoints.

the current usage of IT

It compares which information and communication processes support the core ones and how they are implemented from the manager and the users' points of view. All the available functions are listed and analysed in terms of ease of use, availability, performance, reliability, to characterise the current usage of IT.

The IT manager of the company has to describe as objectively as possible the architecture, current solutions and IT supported functions. This analysis can also be lead by somebody external from the company.

the target usage of IT

During interviews with the managers we collect their objectives in terms of IT policy or new developments. Users outline all opportunities of evolution, improvement or new programs supporting their daily activities.

the potential usage of IT

Finally the diagnosis identifies the most relevant opportunities for introduction of new IT. They result from a cross-analysis of core processes and information and communication ones. The ISO 15504 assessment approach is also tailored to evaluate some of the software/system processes such as customer/supplier relations, project management and risk management.

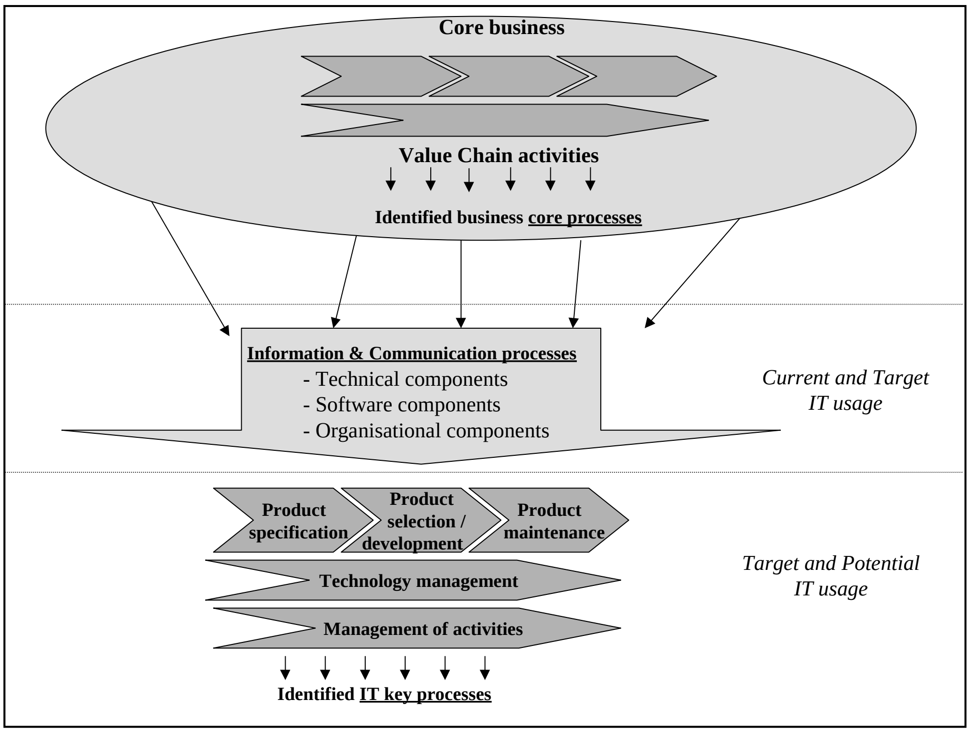

Core business
Value Chain activities
Identified business core processes
Information & Communication processes
- Technical components
- Software components
- Organisational components
Current and Target IT usage
Product specification
Product selection / development
Product maintenance
Technology management
Management of activities
Target and Potential IT usage
Identified IT key processes

Figure BB&AH.1 : Representation of the framework

Conclusion

Some companies of the interregional area have been aware of SPI and are able from now on to initiate SPI projects. The CRP Henri Tudor played and is still playing an important role in spreading software process improvement ideas and initiatives. In order to better apply these concepts and to evaluate the internal usage of IT, the SSD's framework is built. It provides a baseline for working with SSDs, helping them in the management of IT, and more particularly with Information and Communication New Technologies.

Generally speaking, the Centre Henri Tudor is developing the concept of resource centre as a platform dedicated to a sector or a discipline (manufacturing, health care, multimedia, building trades ...). The neutral and independent platform aims to disseminate information and to exchange experiences on software processes, to gather competencies, skills and tools and to propose an integrated set of related services. Under the generic label SPIRAL a variety of activities are organised such as inter-company work groups actively discussing software quality issues, tailored training cycles to create awareness of software process management and ISO 9000 certification among our regional business partners. A SSD's dedicated framework for evaluating the internal usage of IT directly contributes to the development of services in such a resource centre.

References

Porter Michael E., Millar Victor E., How information gives you competitive advantage, in: *Harvard Business Review Article*, July 1985

Kilpi T., Saukkonen S., Luukas J., Nokia Telecommunications Oy, Self-assessment as a Means of Software Process Improvement in Small Software Processing Units, in: *Proceedings of the 3rd annual European Software Engineering Process Group Conference*, London, UK, June 1998

Andrés A., Magnani G., European Software Institute, SPICE for ‘s : How Process Improvement Supports the Achievement of ISO 9001, in : *Proceedings of the 3rd annual European Software Engineering Process Group Conference*, London, UK, June 1998

Brodman G. J., Johnson L. D., The LOGOS Tailored CMM for Small Businesses, Small Organisations, and Small Projects, LOGOS International, Inc., Needham, UK, September 1997

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

Lobet-Maris C., Delhaye R., Henrotte V., Walthery P., Utilisation des Systèmes d’Information Inter-Organisationnels par les PME Belges, in : *Rapport Final SIO*, CITA-FUNDP, Namur, B, November 1997

Compita Ltd, Process Professional Process Portfolio, Process Professional Library Services, 1996

SCATE Mid-term Report - ESSI Project 24291 - MTR, v4, June 1998

TAPISTRY Mid-Term Report - ESSI Project 24238 - V 1.0, January 1998

SPIRE Project Information Pack - ESSI Project 23873 – April 1997

SPIRAL.NET Project Programme - ESSI Project 27884 - V 2.0, May 1998

ISO/IEC JTC 1/SC 7, ISO/IEC TR 15504, 1998

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

EN ISO 9002, Systèmes qualité - Modèle pour l’assurance de la qualité en production, installation et prestations associées, CEN, Bruxelles, B, Juillet 1994

EN ISO 9003, Systèmes qualité - Modèle pour l’assurance de la qualité en contrôle et essais finals, CEN, Bruxelles, B, Juillet 1994

K. Pulford, A. Kuntzmann-Combelles, S. Shirlaw, A quantitative approach to Software Management – The ami handbook, Addison-Wesley, 1995

Kenneth M. Dymond, Le Guide du CMM – Introduction au modèle de maturité CMM, traduction A. Combelles et al. – Objectif Technologie, Toulouse, F, 1997

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").

Philippe LIEMANS

Member of the PRISME project which aims to assist SME's to manage IT. The aims of PRISME are to improve their maturity level to manage the IT infrastructure and to assist them to assess the opportunities of new technologies.

Is in charge to visit SME's and to prepare the diagnosis of their current infrastructure and the analysis of their specific needs.

Jean-Pol MICHEL

Member of the board.
Manager of the Software Engineering Team (35 engineers).
This team gathers all competencies related to design and management of Information Systems. The covered topics are computerisation of operational activities, integration of subsystems, decision support systems, co-operative information systems.
Manager of the training centre of the CRP Henri Tudor named SITec.
This Centre organises continuing training courses, qualification programs and awareness events.

Thanks to all the Software Engineering team members who contributed to this paper.

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.

<u>Main figures</u>

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

CMM in a Micro Team: A Case Study

Joao Batista
ISCAA/CISUC, Portugal

A. Dias de Figueiredo
CISUC, Portugal

Abstract

This paper describes the case study of an application of the Capability Maturity Model (CMM) to a micro team with less than 10 people and very short resources, clearly placed in the first CMM level of maturity, the Initial Level. One major aim of the case, which extended over a period of one year, was to study the extent to which the CMM could be adapted to very small teams. We have concentrated primarily on those parts of the CMM that regard the achievement of the objectives of Key Process Areas (KPA) in Level 2, the Repeatable Level. We have identified some KPAs where improvement could be started, and we have included others as we moved along. Throughout the year we gathered data from a wide range of management and engineering tasks. The results obtained after one year have shown a clear improvement of the process: the number of calls from clients for technical assistance had decreased to 25% of the initial value; the budget requirements had decreased to 28%; and the time devoted to software production had increased by 137%. The pragmatic application of the CMM has thus shown that improvement can indeed be achieved for very small teams. We also came to the conclusion that the model cannot be applied in a straightforward way to such small teams: it must be used judiciously, by constantly adapting it to the environment of the team and by selecting just the key practices that are really relevant to the process. We have also concluded that team management is just as important as process management in this kind of environment.

Introduction

The CMM, or Capability Maturity Model, has been established in response to the so called software crisis of the late 70s, which was a motive of very serious dissatisfaction from the US Department of Defense [1]. This crisis, which has by no means decreased in the software industry as the years passed, results from the lack of maturity of the software development process. Its most visible consequences are the low quality and reliability of most of present day software products.

The bigger problems naturally tended to occur in highly complex systems, for which very large teams had to be assembled. This explains why the CMM considers a team to be small if it is composed of less than 70 people, big if more than 200 people are involved, and medium sized if its dimension falls between those two values [2].

The current software industry is, however, largely made up of very small firms, many even resorting to less than 10 people for software development. We will refer to those as micro teams. Micro teams also have very serious problems of maturity in their software development processes. In fact, in many cases no real process does even exist, often leading to very chaotic modes of operation that affect the whole firm. In the words of Davis, "software is burst of creativity and individual genius rather than teamwork and engineering discipline" [3].

In this paper we describe the case study of an application of the CMM to a micro team. The main objective of the case, besides the obvious one of improving the competitiveness of the team, was to study to what extent we could apply, use, and adapt the CMM to such a small team.

We have chosen the CMM because: a) we view it as the most widely known method for software process improvement; b) it is very well documented; and c) there is a close relationship between the CMM and ISO 9000. Other methods, besides the CMM, that we have not explored include SPR [4], QIP from SEL [5] [6], and SPICE [7].

The CMM [8] [9] [10] recognises a set of five levels of maturity of the software: 1 - Initial; 2 - Repeatable; 3 - Defined; 4 - Managed; and 5 - Optimising.

Each one of those levels expresses a different state of maturity in an organisation devoted to software production. In this scale, level 1 corresponds to the lower state of maturity and level 5 corresponds to the higher state of maturity. We say that the process of an organisation is at a given level of maturity, besides level 1, when the objectives of that level have been attained. The objectives are grouped in Key Process Areas (KPA). Thus, for instance, level 2 will have been attained when all the objectives of the following KPA have been met:

KPA 1 - Requirements Management
KPA 2 - Software Project Planning
KPA 3 - Software Project Tracking and Oversight
KPA 4 - Software Subcontract Management
KPA 5 - Software Quality Assurance
KPA 6 - Software Configuration Management

For each KPA, a number of Key Practices (KP) are defined. The KPs establish what must be realised, but not how it is realised.

In the text that follows, we start by describing the case study and the approach we have taken. We then present results that show that the process has improved in some measure. Finally, we discuss those results and present some concluding remarks.

The Background

The case study is about a small business in which we have applied CMM during 12 months. The business has four units, or sectors: a) software sector (SS); b) commercial sector (CS); c) hardware sector (HS); and, d) finance sector (FS). The SS was a micro team with less than 10 people, with very short resources, and clearly in the first level of maturity, the Initial Level.

The mission of the SS is to develop and supply high quality business software products and services. Clients use software that the SS supplies and the SS make technical support to clients. In return, clients pay a monthly fee.

The SS was very chaotic when the case begun. There was no effective leadership, and motivation was very low. Although significant experience had accumulated over the years, there was no internal or external co-ordination. Technical abilities and practices had poor sophistication. No managerial or engineering procedures or policies had been reinforced, and no administrative support existed.

At the technical level, quality was low. No analysis and requisite management were carried out, version control was poor, no concern existed about reusing code, and no attempts were made to improve quality. Application development was not planned and followed through, and the resources for development were almost exclusively applied in maintaining the existing applications. No one knew exactly what the other people working on.

From the client side, the lack of satisfaction was also very clearly visible. Indeed: a) some of the applications did not carry out all the expected tasks; b) other applications completed the desired tasks, but often in inconsistent manners; and c) support services were neither effective or efficient.

The other sectors of the business also had their own problems. In particular, the negotiation and sale of software service contracts by the CS were based on insufficient knowledge about the corresponding products and services. This often led to last minute changes and adaptations that had not been planned nor managed. The HS did not follow any adequate quality assurance procedures regarding the machines and services they supplied, which often resulted in complaints from clients that put the blame on the SS. Not surprisingly, the financial situation of the firm was very fragile.

All those factors contributed to increase the instability and confusion within the SS.

In other words, software was developed but no process existed to do it. And as no process was identifiable, no repeatability could exist. This clearly put the operation of the SS in CMM level 1, the Initial Level.

The changes described in this case started when a new leadership was appointed for the SS. The main objective was to find out how the CMM could be applied, used and adapted to such a team so that the process could clearly be improved. Particular objectives were: a) to create a repeatable software process; b) to manage the team so as to keep high levels of motivation and performance; c) to prepare the SS for technological evolution; and d) to reason permanently about all items, so that the process could continuously evolve.

To achieve these objectives, the action plan for the first year included: a) the detailed study of the Key Process Areas of level 2, and the clear understanding of their objectives, followed by adaptation and application to the SS; b) the maintenance and correction of the existing applications and support to the clients that owned them; c) the creation of new applications, developed using more advanced technology, so as to induce the analysis and management of requisites, code reuse, version control, and other correct practices; d) the search for new markets; and e) the permanent review of the action plan.

To put this plan into practice, some more resources were needed: a) human resources have been reinforced; b) computers and development software have been upgraded; and c) access to technical information and to the Internet have been granted.

The Approach

The application of the CMM to the SS took place in three phases. The first phase was dedicated to an evaluation of the maturity of the software process. As described above, it was soon found out that the software process was clearly in level 1 of the maturity scale. It was also found out that the only KPA of level 2 that required no intervention was KPA 4 – Software Subcontract Management –, simply because the SS did not resort to software subcontracting. All the other five KPAs were calling for serious intervention, and it soon became clear that it was not practical to intervene simultaneously in all of them.

The second phase of application of the CMM to the SS was concentrated on KPAs 1, 3 and 6. In this way, not only the requisites and configuration of the software products could be controlled, but data could also be collected to describe the behaviour of the SS through the response to questions such as "How long did it take to develop program X and how much did it cost?" or "What's the productivity of programmer Y?".

In possession of data that clarified the time and cost associated to each set of applications it became possible to start establishing development plans. Thus, the third phase could be initiated by concentrating now on KPAs 2 and 5, in search of higher levels of predictability of results for the software process and in questing a more systematic pursuit for quality.

From the detailed application of the KPAs to the SS, which is described in the documentation produced for the case, some methodological aspects should be mentioned here:

the roles established in the CMM have been simplified, given the dimension of the team;

the norms and procedures have not generally been written down, but rather transmitted and interiorised by the group at the appropriate times;

all the elements of the SS actively participated in the resolution of any problems and in determining the direction of the SS, both at the technical and management levels;

the KPAs and their KPs have been evaluated keeping in mind the need to apply the CMM with pragmatism: a) because the team was small; and b) because the resources were very scarce.

Given their particular relevance for success, some specific aspects of the application of the CMM to our SS should also be stressed:

software requisites and their changes were always written down and always resulted from a process of analysis;

the times and costs associated to the various activities have been measured and registered. Each element of the team filled up a daily time sheet where the tasks carried out were specified in agreement with a pre-defined table, and the starting and finishing times were registered. Those tasks could be merely technical, such as "programming", in which case they were associated to an application, or they could be management tasks, such as "technical support", in which case they were associated to an application and a client. After some "tuning" time, an application has been developed to register those elements automatically;

the systematic use of metrics led, after some time, to creation of expectations (though not yet estimates) for the development of the projects. This initial process of "expectation management" will soon became a more rigorous process of: a) estimation of the size of the software solutions; b) estimation of the costs involved; and c) estimation and monitoring of the calendar for each project;

monitoring and supervision became a systematic task of the team leader, who made sure that changes and improvements were always agreed with the people involved and made known to the whole team;

development plans for each software project started to be established, though not fully written down. Each project was regularly discussed with the elements of the SS before tasks were distributed and resources assigned to each task. The team leader was responsible for obtaining the required resources in negotiation with the other sectors of the firm;

the SS accepted the commitment to a permanent search for quality, and as the small size of the team made it impossible to set up a quality assurance group, some general practices started to be followed instead. For instance, the systematic revision of a product or sub-product was carried out by people that were not involved in the development of that particular product or sub-product. In this way, the whole SS became responsible for the quality processes. Though this did not grant real independence in the quality management process, it was found to be the feasible and acceptable compromise;

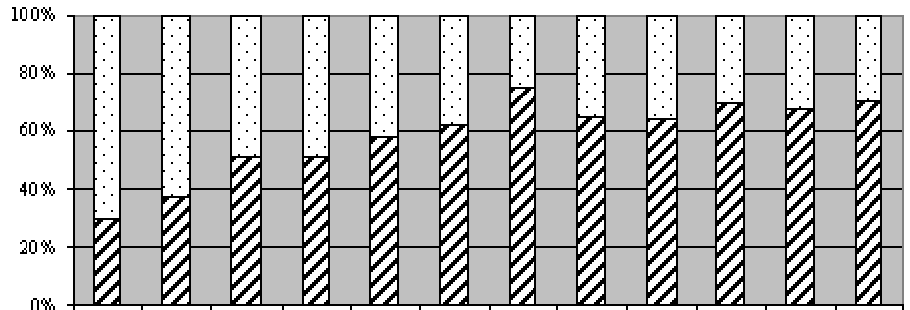
the SS recognised the need for a rigorous control of all versions, and applications and utilities have been developed to build up a digital repository of some of the elements and to manage the placement and life cycle of those elements that were not available in digital form.

The Results

After the first year of application of the CMM many results could be gathered and organised. The most significant ones are shown.

Time Results

It is clearly noticeable that throughout the period of analysis the time dedicated to software production (which corresponds to the development of new applications and the maintenance of existing ones) has increased. Conversely, the time dedicated to other activities, which we described as non-production activities, has decreased. In this category, the activity with most weight was technical support to clients. Fig. JBADF.1 shows that: a) the percentage of monthly time dedicated to software production has increased 137% throughout the 12 months; and b) the percentage of monthly time dedicated to non-production activities has decrease to 43% of its initial value.



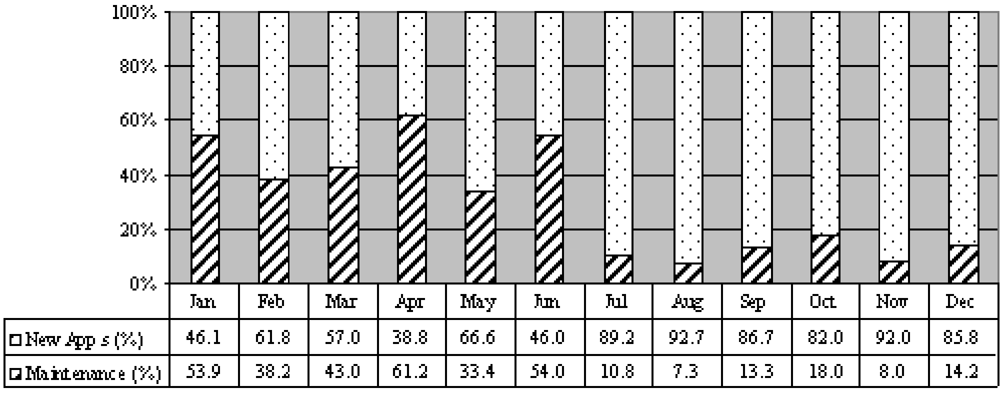
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
□ Non-Prod. (%)	70.6	62.7	48.7	48.8	42.5	37.9	25.0	35.4	36.5	30.5	32.6	30.2
▨ Production (%)	29.4	37.3	51.3	51.2	57.5	62.1	75.0	64.6	63.5	69.5	67.4	69.8

Fig. JBADF.1 - Time spent in the activities of the software sector (SS): time dedicated to non-production activities versus time dedicated to production activities.

Those results are reinforced by the comparison between the time dedicated to the maintenance of existing applications and the time spent developing new applications. As can be seen from Fig. JBADF.2, the percentage of monthly time dedicated to the maintenance of existing applications has decreased to 26% of its initial value, while the time devoted to the production of new applications has increased by 86%.

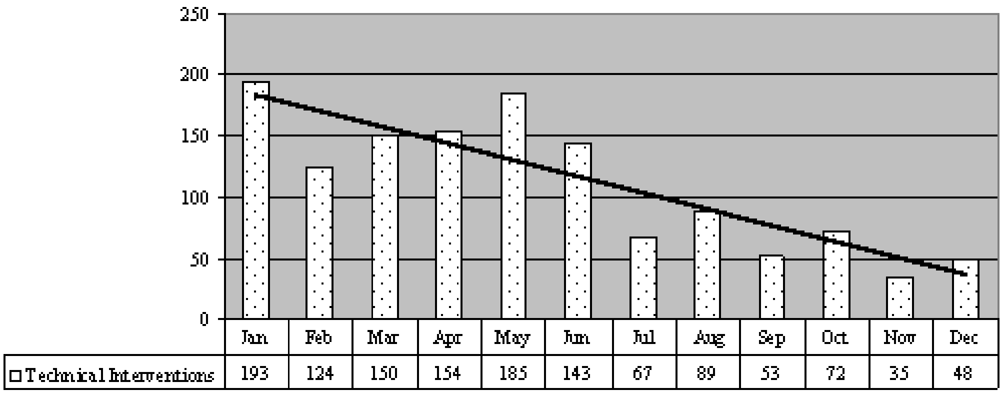
Client Results

From Fig. JBADF.3, we can see that the number of monthly interventions in response to client calls has been reduced to 25% of its initial value. During the period of our analysis the number of clients increased slightly, but not significantly. Some clients have been lost during the first months, but new ones were brought in the meantime.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
□ New Apps (%)	46.1	61.8	57.0	38.8	66.6	46.0	89.2	92.7	86.7	82.0	92.0	85.8
▨ Maintenance (%)	53.9	38.2	43.0	61.2	33.4	54.0	10.8	7.3	13.3	18.0	8.0	14.2

Fig. JBADF.2 - Time spent in the activities of the software sector (SS): time developing new applications versus time maintaining existing applications.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
□ Technical Interventions	193	124	150	154	185	143	67	89	53	72	35	48

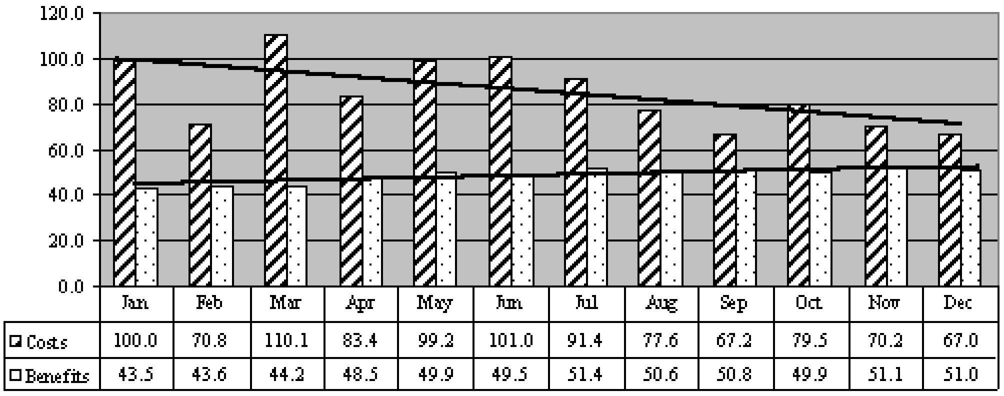
Fig. JBADF.3 - Number of technical support interventions for clients.

Financial Results

To express the financial results that follow an index system is used instead of real values. The total cost for the first month is expressed as index 100 and all the others are compared with it.

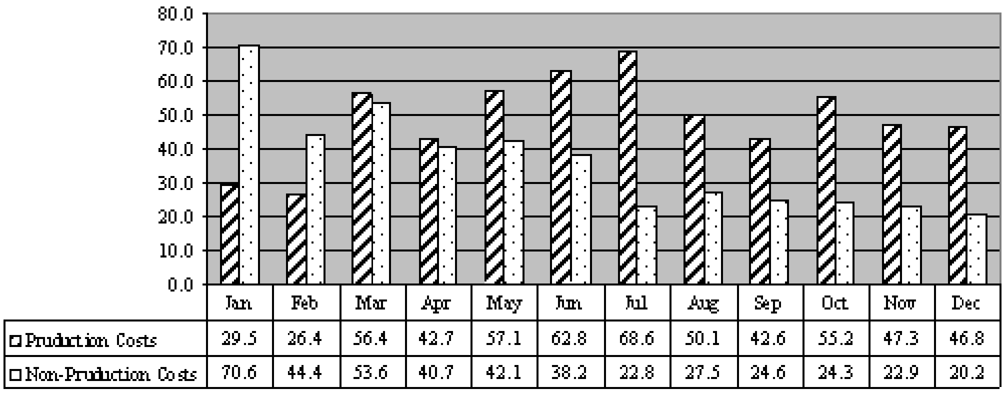
From Fig. JBADF.4 we can see that: a) monthly costs decreased to 67% of their initial value; b) monthly benefits increased 17% of their initial value; and c) the difference between benefits and costs has decresed to 28% of its initial value, nearing zero and suggesting a trend towards positive values.

Fig. JBADF.5 illustrates monthly costs discriminating production activities and non-production activities. It shows that: a) the monthly cost of production activities has increased 59% throughout the 12 months; b) the monthly cost of non-production activities has decreased to 29% of its initial value; and c) those results are consistent with those presented in Fig. JBADF.1.



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
▨ Costs	100.0	70.8	110.1	83.4	99.2	101.0	91.4	77.6	67.2	79.5	70.2	67.0
□ Benefits	43.5	43.6	44.2	48.5	49.9	49.5	51.4	50.6	50.8	49.9	51.1	51.0

Fig. JBADF.4 - Costs and benefits of the software sector (SS).



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
▨ Production Costs	29.5	26.4	56.4	42.7	57.1	62.8	68.6	50.1	42.6	55.2	47.3	46.8
□ Non-Production Costs	70.6	44.4	53.6	40.7	42.1	38.2	22.8	27.5	24.6	24.3	22.9	20.2

Fig. JBADF.5 - Production activities costs versus non-production activities costs.

Discussion and Conclusions

The results we have presented are globally consistent between them. The SS holds a high degree of autonomy, and during the 12 months of the case study it has not been subject to any new variables. Thus, the results suggest that the application of the CMM to the SS has contributed significantly to process improvement.

Some doubts may arise regarding the interpretation of the financial information in Fig. JBADF.4, since the trend in the reduction of costs is much more evident than the increase in benefits. We believe that this resulted from the rigour and discipline that the use of the CMM has imposed within the SS. On the other hand, the increase in benefits has depended strongly from the development of new applications, which took some time, and of the subsequent profitability. We believe that, as time goes by, the trend line for costs will tend to stabilise, or even grow up, and the benefits line will tend to grow more sharply.

The application of the CMM to the SS was by no means easy. Given the small dimension of the team and its lack of resources, many simplifications of the method had to be introduced, and only the KPs that were really important for the process have been retained. We could say that the CMM has been applied in a very pragmatic way.

The attempt to move from level 1 to level 2 of the maturity scale has not been completed. At present, the maturity of the SS lays above level 1 but below level 2.

In general, we may conclude that:

the pragmatic application of the CMM to the SS has led to significant improvements that put its software process above level 1;

the software process is below level 2, but showing that this level can be achieved;

the application of the CMM to micro teams is possible and contributes to the improvement of their software process; however, simplifications of the method are required, namely in the structure of the functions and in the formalism of procedures and norms; the costs of trying to fully apply the CMM to micro teams such as the one described would be far too high;

the results obtained are globally positive and consistent between them, which suggests that they are a consequence of the pragmatic application of the CMM;

the pragmatic application of the CMM leads to higher levels of quality of the resulting software;

the CMM is exclusively concerned with the software development process.

The application of the CMM to a micro team has shown that another critical factor must be taken into careful account besides the software process: the human resources factor of the team and its management. This has shown to be particularly sensitive, because: a) the level of maturity of the SS was initially very low; and b) the small dimension of the SS did not facilitate a global cultural change, unless it was attempted directly through each member. In summary, the application of the CMM to a micro team must be carried out judiciously by permanently adapting to the environment of the team, using just the KPs that are really important to the process, and keeping in mind that team management is just as important as process management.

The relevance of the factors regarding human resources has been recognised by other authors [11], and even by the SEI in its publication of CMM V.1.1: "The CMM may also become multi-dimensional to address technology and human resource issues" [8]. This resulted, in 1995, in the publication of the P-CMM - People Capability Maturity Model [12]. Also in 1995, the technological dimension has been addressed in the SE-CMM - Systems Engineering Capability Maturity Model [13]. These are factors that must be taken into further account in future refinements of the micro team approach, so as to make the application of the CMM more consistent with the dimensions of process, technology and human resources.

Acknowledgements

This work has been partially supported by the Portuguese Foundation for Science and

Technology (FCT) under research contract 326/94. The authors are grateful to ISCAA and CISUC for all the facilities granted.

References

[1] Humphrey, W., Characterizing the Software Process: A Maturity Framework. *IEEE Software 5*, 2 (March, 1988), 73-79

[2] Goldenson, D., and Herbsler, J., *After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors that Influence Success*. CMU/SEI-95-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 1995

[3] Davis, A., and Leffingwell, D., *Using Requirements Management to Delivery of Higher Quality Applications*. Rational Software Corporation, U.S.A., 1996

[4] Jones, C., *Assessment and Control of Software Risks*. Prentice-Hall, Englewood Cliffs, New Jersey, U.S.A., 1994

[5] Basili, V., and Green, S., Software Process Evolution at the SEL. *IEEE Software 11*, 4 (July, 1994), 58

[6] Basili, V., Zelkowitz, M., McGarry, F., Page, J., Waligora, S., and Pajerski, R., SEL's Software Process-Improvement Program. *IEEE Software 12*, 6 (November, 1995), 83

[7] Emam, K., Drouin, J.-N., and Melo, W., (editors) *Spice: The Theory and Practice of Software Process Improvement and Capability Determination*. IEEE Computer Society Press, Los Alamitos, California, U.S.A., 1997

[8] Paulk, M., Curtis, B., Chrissis, M., and Weber, C., *Capability Maturity Model for Software, Version 1.1*. CMU/SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 1993

[9] Paulk, M., Weber, C., Garcia, S., Chrissis, M., and Bush, M., *Key Practices of the Capability Maturity Model for Software, Version 1.1*. CMU/SEI-93-TR-25, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 1993

[10] Humphrey, W., *Introduction to Software Process Improvement*. CMU/SEI-92-TR-7, revised, 1993, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 1993

[11] Bach, J., Enough About Process: What We Need Are Heroes. *IEEE Software 12*, 2 (Março, 1995), 96

[12] Curtis, B., Hefley, W., and Miller, S., *People Capability Maturity Model*. CMU/SEI-95-MM-02, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 1995

[13] Bate, R., Kuhn, D., Wells, C., Armitage, J., Clark, G., Cusick, K., Garcia, S., Hanna, M., Jones, R., Malpass, P., Minnich, I., Pierson, H., Powell, T., and Reichner, A., *A Systems Engineering Capability Maturity Model, Version 1.1*. CMU/SEI-95-MM-03, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A., 1995

Joao Lopes Batista is an Adjunct Professor of Informatics at the Higher Institute of Accounting and Administration of Aveiro (ISCAA) since 1987. He obtained his Master's Degree in Informatics from the University of Coimbra in 1993, and he is currently working on his Ph.D. His main research interests concentrate on Strategy and Management for Information Systems, Software Engineering and Object Technology. His past professional activities included lecturing at the University of Aveiro as an invited professor, and he has been a consultant for external companies in the fields of Software Engineering and Management. He is a researcher of the Centre for Informatics and Systems of the University of Coimbra (CISUC) since 1994. He is a member of the ACM, of the IEEE, and of the Portuguese Professional Institution of Engineers. He is the author of several papers, communications and research reports.

António Dias de Figueiredo is a Full Professor of Informatics Engineering at the Faculty of Science and Technology of the University of Coimbra, Portugal, since 1984. He obtained his Ph.D. in Computer Science from the University of Manchester, U.K., in 1976. His main research interests concentrate on Strategy and Management for Information Systems, Software Engineering, and Information and Communications Technologies in Education. He is the doyen of the Department of Informatics Engineering of the University of Coimbra, which he founded in 1994 and chaired until March 1997. He is the doyen and founder of the Centre for Informatics and Systems of the University of Coimbra (CISUC), the institution for R&D in Informatics of the University of Coimbra. He represents Portugal in the Intergovernmental Informatics Programme of UNESCO, where he acted for two years as Vice-president elected for the Western Europe Region. He has participated in various European projects, both as a partner and as a science advisor, and acted in various occasions as a consultant to the European Commission in matters regarding the definition of strategies for information and communications technologies in education. At the invitation of the NATO Science Committee, Brussels, he integrated, for a period of four years, the NATO Special Programme Panel on Advanced Educational Technology. At the invitation of the European Commission, he contributed to the preparation of the White Book on Education and Training for the XXI Century. He is a member of the Panel for Research and Development in Consortium of the National Innovation Agency, and integrated various international panels for the approval and evaluation of research projects. He is a member of the panel of the IBM Science Prize since its creation in 1989. He is member of the Portuguese Engineering Academy. He is a member of the ACM, of the Portuguese Engineering Academy, and of the Portuguese Professional Institution of Engineers.

He is an elected member of the Council for Qualification and Admission of the Portuguese Professional Institution of Engineers. In 1997 he was awarded an *Honoris Causa* by Universidade Aberta, the Portuguese Open University. He is the author of over 120 papers and presented over 140 communications. He has integrated over three dozens organising and science committees of conferences held both in Portugal and abroad.

The **Instituto Superior de Contabilidade e Administração de Aveiro**, ISCAA, (Higher Institute of Accounting and Administration of Aveiro) is a well-established school of Accounting and Administration of the Portuguese network of public polytechnic institutes. It has been created in 1966, and is currently attended by a population of 1400 students taking their graduations in "Accounting and Auditing" and in "Business Administration". In association with the Portuguese Open University it is running a Master's degree in "Accounting and Business Finances". It has collaboration protocols with the universities of Sceaux and Brighton within the Erasmus Program. It has a staff of about 60 professors and lecturers deeply involved in research projects within their areas.

The **Center for Informatics and Systems of the University of Coimbra** (CISUC) is a large Portuguese research institute in the fields of Informatics and Communications, which has been created in 1991 with the aims of carrying out R&D at a pre-competitive level, training highly qualified professionals, co-operating in national and international projects and programs, and promoting the dissemination of results, namely through contracts with national and international companies. It is currently composed of eight research groups in the areas of Software Engineering and Information Systems, Artificial Intelligence, Communications and Telematic Services, Simulation and Technologies in Education and Training, Control Theory, Dependable Systems, Combinatorial Optimization, and Theoretical Fundamentals of Computer Science. Most of its research activities are developed within national and international projects and research networks, with the support of funding programs, such as PRAXIS XXI and ESPRIT, thus contributing to the R&D effort of the European Community. Its scientific results are regularly published in well-established journals and presented in prestigious conferences. The quality of its work is expressed in national and international awards that the members of the CISUC have received since its foundation, their participation in the scientific and editorial boards of conferences and journals, and their membership to government and professional committees responsible for the definition of R&D policies. Interaction with industry and other private and public institutions, national and international, is strengthened through Instituto Pedro Nunes, a non-profit private organization founded in 1991 for assisting economic agents in their efforts towards global competitiveness.

Improving Estimation and Requirements Management: Experiences from a very small Norwegian Enterprise

Svein Are Martinsen (sveinare@invenia.no)
Invenia AS, P.O. Box 282, N-9201 Bardufoss, Norway
Arne-Kristian Groven (groven@nr.no)
Norwegian Computing Center (NR), P.O.Box 114 Blindern,
N-0314 Oslo, Norway

Invenia AS is a very small Norwegian software development organisation with ten developers, located in Northern Norway. We are currently performing an ESSI Process Improvement Experiment (PIE) funded by the European Commission. The aim has been to improve requirements management in general and estimation in particular, focusing on object-oriented modelling in this process.

Analysis of previous development projects in the company revealed some weaknesses. Estimation had previously been performed using expert judgements, taking some experiences into account. Huge financial losses and schedule overrun were common on fixed-priced contracts based on poor estimates without proper knowledge of what to develop. General, yet unspecific and poor, documentation made it easier for the customers to claim more functionality.

New routines, addressing these problems, have been defined and have been implemented in two development projects involving 3-5 developers. The new routines are supported by a tool for documenting and analysing requirements, and are used in combination with an object-oriented modelling tool. Use case models are now the analytical basis for estimation of more complex entities.

The experiences with the software process improvement experiment have been positive. This paper describes the plan and implementation for our software process improvements as well as the results and lessons learnt from the experiment. We also consider specific challenges for small organisations when improving their software process.

Invenia AS

Our company, Invenia AS, is a small Norwegian enterprise with 14 employees, 10 of which are producing tailor-made software (client/server) for customers, often on a fixed-price basis. The business idea of the company is to help customers realise their business goals through strategic IT-consultancy and development of software solutions supporting the business processes of the customer's value chain.

Invenia AS specialises in client/server and collaboration technology for Windows, Windows NT, and the Internet. Development tools include MS Visual Studio (Visual Interdev, Visual Basic), MS FrontPage, C++, and PowerBuilder. As a server platform we focus on Windows NT with Internet Information Server, MS Exchange, and relational databases such as MS SQL Server, Sybase SQL Server, and Sybase SQL Anywhere.

All developers have university or college education, and have work experience ranging from 0 to 8 years.

Starting Scenario

Invenia's management experienced a growing dissatisfaction with respect to estimation in particular and requirements management in general [1, 2, 3]. This resulted in an application for fund within the European Systems & Software Initiative (ESSI), and the application was accepted. The result was a Process Improvement Experiment (PIE) called IMPOSE, ESSI PIE project number 23780 [12, 13].

The SPI project started in 1997 and is planned to be finished by December 1998. Before going into details about the SPI approach itself, we shall first create an initial setting. We will begin with a brief description about the organisational context and direction, prior the SPI experiment.

Organisational Orientation towards SPI

Invenia AS has experienced an increasing awareness and orientation towards software processes and software process improvement. This started some time before the SPI experiment about which we are going to report. The management considered software development process control as one of the most important factors by which to secure the company in the long term. However, experience also showed that this wasn't always easy, due to the continuously changing technological environment.

In 1995, the company introduced an iterative software process model which can be described as follows: From an initial project description, the phases of analysis, requirement specification, design, implementation, and testing were repeated, typically three or four times in a nine month project. There were different goals as main milestones for each iteration, shifting focus gradually from analysis and requirement specification towards implementation and testing.

The process model had been used in larger projects, while smaller projects were performed more ad hoc. Experience with the process model was positive in many senses. However, neither the management nor the development team was yet satisfied with the results achieved using this approach. Far too often, software products were delivered too late, with a too high internal cost, since more resources than planned were needed to finalise them. This condition clearly indicated lack of sufficient control.

The company started to analyse the problems related to the implementation of their software development process a bit further. The findings were as follows:

The estimates were usually made in the initial project description, which seems to be a poor basis for cost estimation.

The requirement specification was only loosely coupled with the effort and cost estimates. The feasibility of realising the requirements within cost limits was not judged closely enough.

The requirement specification was written in natural language, and was too informal to be a good basis for a necessary revision of the cost estimate or a good basis by which to restrict the implementation within the cost estimate.

Analysis models were not used enough for communicating requirements with the customer/users, and the difference between analysis and design was not well defined.

The milestones of each iteration were too inaccurate. Too often, there was a problem deciding when an iteration was completed.

Changes in the implementation were not automatically reflected in analysis and design documents. They were costly to maintain by hand.

During this period, key personnel had also taken part in both regional and national IT-networks, including Software Process Improvement networks. Here, experiences were presented and contacts were made, among others to SPI professionals. These contacts eventually resulted in the ESSI PIE project.

Analysis of Estimate Overrun

The company had been gathering project data, which turned out to be very useful. This allowed the SPI initiative to start by analysing a reasonably detailed data set of all of the company's fixed price projects in the period between 1993 and 1996.

The results showed that none of the development projects had produced accurate estimates; more precisely, there were estimate overruns for all the projects. The following table illustrates the estimate overruns relative to project size, either small, medium, or large:

Project size (estimated)	*Average overrun*	*Best-project overrun*	*Worst-project overrun*
< 100 hours	163.6 %	7.3 %	406.7 % (!)
100-1000 hours	30.9 %	9.8 %	53.3 %
> 1000 hours	74.0 %	19.9 %	161.7 % (!!)

Fig. SAM.1: Estimate overruns for previous projects

In general, estimation and scoping are a lot more difficult within a heterogeneous, evolving technology environment. Our software developing company is a typical example of such. Some estimation problems could also specifically be related to project size and the different nature of small projects compared to larger ones.

All projects were vulnerable to so-called "scope creeps". That is: " The propensity of the scope of application development projects to increase as the project proceeds through its life cycle " [4]. In the smaller projects, this was possible due to lack of proper analysis up front, combined with minimal project documentation. Agreements with customers were signed, based on vague statements about the products to be developed, making it easy for the customer to require more than he/she actually had paid for. In the larger projects, analysis was performed prior to estimation. Here also, estimation accuracy was low, initiating a search for better solutions. Additional factors regarding project and product complexity also influenced the poor results.

Most alarming was of course the situation for the largest fixed price development projects. However, it was a clear tendency towards decreased estimate overrun from one project of that size, to the next. Still, an approximate 20% overrun as in the most recently analysed large-size project is neither satisfactory nor acceptable.

Here, it is necessary to add a few words about the medium-sized projects. They were usually performed by, one sometimes two, developers. On the one hand, they had more available time to perform the work, compared to the smallest projects. On the other hand, they had less complexity than larger projects (e.g., with regard to project communication and number of technologies involved. Together, these factors might have had a positive effect upon estimation.

In the largest projects, there was a tendency towards involving too many new development technologies within a short period of time.

Plans and Expected Outcomes

The initial planning of the IMPOSE project took place half a year before it was accepted by the European Systems & Software Initiative (ESSI). After acceptance, an official plan document was developed, and the project was ready to start some months after that.

SPI Goals

The main goal of the SPI experiment was:

Accurate estimation and scope control.

In order to achieve this goal a more detailed set of subgoals was formulated, based on the analysis of the past projects. Each of these subgoals serves as a mean for achieving the main goal, based on the underlying hypothesis that they are critical success factors. Below, an overview of these subgoals is given.

Techniques for requirements elicitation and analysis [1,2], including formal analysis models, should be used in the initial phases of the software development process to improve the understanding of the problem domain.

In particular, formal analysis models should be used more actively when communicating with end users, in order to establish a common understanding of the problem domain.

The result of the requirements analysis should be a stable set of requirements that is documented in a requirement specification document and is used as a basis for further development activities [3].

Such models can be achieved by using object-oriented techniques [5, 6]. Thus, guidelines for object-oriented analysis and design, encouraging activities leading to a more complete object-oriented design ready for implementation, have to be established.

The new software process should support an iterative approach for improving the effort and cost estimates during the (first phases of the) development life cycle, based on better understanding of the problem domain.

New and improved estimates should finally be based on the requirement specification document, possibly resulting in a revision of the legal agreement with the customer should (substantial) deviations occur.

The first three points cover the topics referred to in literature as requirements engineering and requirements management. The fourth point specifically mentions use of object-oriented techniques. This is of particular importance regarding the agreement between ESSI and the company. The last two points have to do with estimation, or rather the revision of estimates over time, based upon better understanding of the problem domain (through improved requirements engineering). This point can be illustrated as follows:

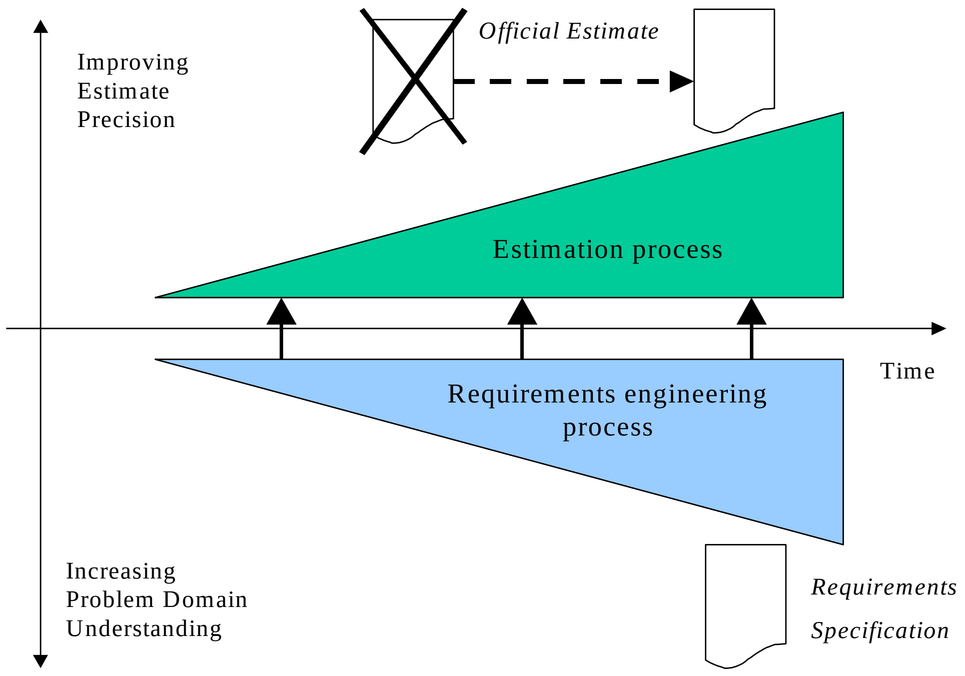

Improving Estimate Precision
Official Estimate
Estimation process
Time
Requirements engineering process
Increasing Problem Domain Understanding
Requirements Specification

Fig. SAM.2: Estimation and requirement refinement

The aim is not only to achieve better understanding of the project scope, but also to document this understanding and to help better motivate the customer towards the joint development of a better product. All requirements shall be documented in a requirements specification document, and finalisation of the official estimate — the basis for the legal agreement with the customer— shall not take place until a stable set of requirements exists. Time is critical in this process. A customer will not accept too long time frame from project initialisation, until an finalisation of estimates. The challenge is not to compromise quality for time.

The company's success criteria for the experiment was stated in quantitative terms as follows:

30% reduction of total overrun time;

20% reduction of financial loss caused by overrunning cost estimates and missing milestones.

Like the effort estimates described thus far, the schedule estimate overruns have also been considerable in size. In this paper, we have chosen not to go into many details about the overruns in the schedule estimates.

SPI Plan and Activity Overview

The IMPOSE Project Programme, the official plan document for the project, defined the following set of work packages:

WP1: Improve existing and define new methods and routines
WP2: Evaluate and select supporting tools

WP3: Organisational and technical implementation of tools
WP4: Training in use of new routines and supporting tools
WP5: Evaluate baseline project at milestones
WP6: Report results
WP7: Dissemination
WP8: Project management

The project was originally planned as the following sequence of activities:

analyse weaknesses and prepare for process change by defining improvement goals, how to reach them and tool support (WP1-4);

implement new process routines in one regular development project called the baseline;

take measurements, during and after the baseline, and evaluate if it represents an improvement (WP5).

The IMPOSE project was planned to be implemented as one iteration of the activity sequence above. This was done many months before the scheduled start of the baseline. But early in the project it became clear that the originally planned baseline, scheduled for nine months, had to be cancelled. This resulted in an intensive search for other suitable development projects as baseline, and in addition an extensive replanning at an early stage.

After assessing the risks, it was considered to be easier to run the SPI project as two iterations. This was done because of the uncertainty about suitable baselines and baseline lengths. It also seemed preferable not to make all the process changes at once, but instead focus on some of the changes when running the first baseline and then adjust and augment the first approach when running it on a second baseline. This seemed to be a wise decision for the SPI experiment, but it generated too many reports to ESSI.

Implementation of the Improvement Actions

An upcoming project was selected as the first baseline. Due to tough customer time constraints, it had to start before all preparations regarding process, technology, and training could be completely finished. Therefore it was natural to focus on what was considered to be the most important process changes and supporting tools. By providing only light tool support, it was possible to train the development team in both process and tool support before running this first baseline project.

Based on the experience from the first baseline, further improvements including more focus on tool support was considered in the second iteration.

The first Iteration

During the start of the first iteration, a lot of effort was put into improving the software process. This was based on the broad study of previous projects as referred to earlier in this paper. During this work a need for establishing a more documented software process was identified. Previously, each development project had actually spent a significant amount of resources just in defining how to run it. The actual process varied from project to project, and was very dependent on the project leader.

The most important work of the first iteration was:

analyse previous projects;
gain knowledge of software process improvement, object oriented analysis and design, and estimation;
define how and when to estimate;
define how to find the basis for estimation (object oriented modelling);
suggest changes and describe the resulting process model;
specify requirements for, evaluate and select supporting tool for object oriented modelling;
brief training of developers;
run the baseline project;
evaluate the baseline project.

Two senior developers did most of the work in defining the new process. This was positive for the SPI activity in many ways: They had e.g. personal experience with the company practise over a period of time, and also project management experience. Also, they had clear vision regarding where to start changing the process. A consultant from the Norwegian Computing Center was also involved. In addition, all the developers of the company were involved at a couple of meetings to be updated and to provide feedback.

Estimation and requirements elicitation, analysis, and documentation were the main focus. The most important idea behind this estimation approach was to improve estimate precision by gradually gaining better understanding of what to develop during the project [7]. A requirements document was no longer a brief, incomplete, ambiguous description of what the customer wanted. The requirements document was the result of eliciting, analysing, and prioritising the requirements together with users/customers.

The new requirements document should first of all describe how to meet the customer/user requirements, i.e. the (main) functionality. This description should be done through use case and class modelling. By doing so, design activities were actually moved into an earlier phase. Estimation was performed, use case by use case or class structure by class structure. Refinements of use cases should result in refinements of estimates. Each use case was also prioritised, and analysed regarding other attributes. A contract should not be signed until a stable set of analysed requirements existed.

Regarding tool support, a tool for object-oriented analysis and design was most needed. The Rational Rose tool [8] was selected in competition with a handful of similar tools. We also briefly considered the need of tool support for managing and evaluating the requirements, but decided to postpone this to the next iteration, and use a more manual approach with a simple MS Excel spreadsheet for this purpose.

The table below summarises the most important activities and tool support in the defined process model. The focus in the first baseline was on the first two phases.

Phase	Important activities	Tool support
Pre-study	Problem definition Customer analysis Brief use case modelling Identifying important business objects Rough estimation based on top level use case and important business objects	Rational Rose for OO modelling, MS Word for documentation, MS Excel for estimation, MS Project for planning
Requirement specification	Customer analysis Eliciting requirements through use case modelling and describing user scenarios Documenting user requirements (as use cases) Object oriented design; focus on business objects High level user interface design Prototyping Requirements analysis and evaluation regarding attributes like complexity, priority, size, implementing technology etc. Bottom up estimation of needed effort to implement requirements	Rational Rose for OO modelling, MS Word for documentation, MS Excel for evaluating requirements and estimating, MS Project for planning
Development	More detailed design Database design and implementation Development, part by part Module testing	Rational Rose, Development tools, MS Excel, MS Word
Testing	System integration testing Error correction Delivery, ready for deployment	Development tools, MS Excel, MS Word

Fig. SAM.3: Preliminary process model (as defined in first iteration)

The two developers in charge of IMPOSE participated in an external course in object-oriented analysis and design with the Unified Modeling Language [6, 9, 10], before organising a brief training period in the new process and supporting tool.

Due to customer requirements, the project was covered by 3 contracts: one for the pre-study phase, one for the requirement specification phase, and one which should cover the development and testing phases. As there were tough negotiations with the customer regarding the finalisation of the product, the basis for estimation was re-iterated three times. Although the customer was very satisfied with the work done on developing a requirement specification with associated models, he also negotiated with another supplier based on the requirements specification developed by Invenia. The final offer from the customer was not accepted by Invenia, mostly due to the detailed knowledge gained by using the new process. Without this knowledge the company would probably have accepted the project by underestimating the effort.

Invenia was in general satisfied with the new methods and tool support. Although the project was not completed through all four phases, the general opinion was positive regarding the work in the two initial phases. The first brief pre-study phase was positive in order to elicit the concepts and to get an overview of the size of the project before detailing the requirements in the next phase. The detailed modelling prior to estimation provided a less ambiguous understanding of the requirements and how to build the system. Also, customer and developers had a common, documented understanding of the requirements, and during the requirement specification phase several changes in requirements were revealed. Such changes were previously often revealed later during development, when contracts were unchangeable.

Because of the cancellation, it was impossible to test the produced estimates. Although the customer later have told us that “it is no secret that the other supplier missed on their estimate”, there are no useful quantitative results from the first baseline. Therefore, we were really excited about continuing with the second iteration.

The second Iteration

Based on the experiences from the first iteration, the software process was somewhat adjusted. During that work we became aware of the Microsoft Solution Framework (MSF) [11], which seemed to have interesting similarities with the software process model used in the first baseline. The impression was that the more standardised MSF could make further improvements and training easier. MSF also presented interesting ideas for establishing a shared vision, selecting focus early in the project, as well as other issues, e.g. a more consistent set of terms, activities and deliverables. Iterations over conceptual, logical and physical design during the second phase also seemed useful as it encouraged iterative estimates. The results from the first baseline were therefore adapted into MSF.

The most important work of the second iteration was:

adopt to the process model of MSF;
make improvements according to experiences of the first iteration;
specify requirements for, evaluate, and select software tools for document automation and requirements management;
document the new software process, in particular with regard to integration of the supporting software tools;
more comprehensive training in the new software process and supporting tools;
run the second baseline project;
evaluate the second baseline project.

The four phases of the iterative process model are now (according to MSF) called envisioning, planning, developing and stabilising. However, a major effort was put into detailing the process model according to *the company's* needs and experience from the previous baseline, focusing on using object-oriented analysis and design to produce an improved estimation basis.

During the first iteration, a need for more efficient information sharing was identified, combined with more written documentation and reviewing. In the first baseline, too many informal meetings had taken place. How to improve documentation by more efficient use of Rational Rose and MS Word in combination, was of particular interest. A search for additional software tool support to help automate this work also took place, resulting in the selection of Rational SoDA.

Another experience from the first iteration was that managing and evaluating requirements manually was time consuming and challenging. Requirements for a supporting tool were therefore defined, and several tools were evaluated. A tool called Rational Requisite Pro was finally selected. Two senior developers went through a brief self-study in using the tools on small-scale examples. Then the tool was integrated into the process, by describing how to use the other tools and methods together with Rational Rose in order to elicit, document, and manage the basis for cost estimates. We also defined SoDA templates to automate the production of requirement specification documents and a "standard" Requisite Pro project template with requirement types and attribute definitions according to the company needs.

Before starting the second baseline project, the three developers involved were trained in the new software process, in using Rational Rose for object-oriented modelling with the Unified Modelling Language, and in using Requisite Pro for requirements management and evaluation. The training was done more extensively than for the first baseline project. Worth mentioning is that the project manager in this baseline was without project management experience.

The table below summarises the most important activities with tool support for each of the four phases of the new software process.

Phase	**Important activities**	**Tool support**
Envisioning	Problem definition Establishing a shared vision Customer analysis Defining a solution concept through brief use case modelling and high level class design (important business objects) Rough estimation based on top level use case and important business objects Brief risk assessment (managing top ten risks) Rough planning of the project	Rational Rose for OO modelling, MS Word for documentation, MS Excel for estimation, MS Project for planning
Planning	Customer analysis Eliciting and documenting requirements through use case modelling (scenarios) OO design; focus on business objects Generate requirement documents Initial requirements analysis and evaluation regarding attributes like complexity, priority, size, responsibility, required skills, etc. Initial bottom up estimation of needed effort to implement requirements High level user interface design Prototyping Further use case modelling (more details)	Rational Rose for OO modelling, MS Word and Rational SoDA for documentation, Rational Requisite Pro for managing and evaluating requirements, MS Excel for estimation, MS Project for planning

	Detailing use cases with textual descriptions and/or object interaction diagrams illustrating important scenarios. More logical design, defining structure (business, database, and user interface objects) Regenerate requirement documents Requirements analysis and evaluation Bottom up estimation of needed effort to implement requirements If needed, the logical design is evaluated and estimated similarly Brief risk assessment (managing top ten risks) Planning the rest of the project	
Developing	More detailed design (physical) Database design and implementation Development, use case driven Module testing Estimate tracking	Rational Rose, Development tools, MS Excel, MS Word
Stabilising	System integration testing Error correction Delivery, ready for deployment Estimate tracking	Development tools, MS Excel, MS Word

Fig. SAM.4: The new process model (as defined in the second iteration)

Again, the focus was mainly on the two first phases. As shown in the table, eliciting, analysing and estimating requirements (conceptual design) should be done at least twice. If needed several iterations over these may be done.

Compared to the previous baseline project, both information sharing and requirements management was far more efficient during the planning phase. This was due to the tool support introduced.

During the development phase, a database design was completed and implemented before the rest of the system was implemented use case by use case. Each developer recorded the actual work at use case level. Approximately half way into the development phase, the company upgraded the time tracking system. Also, the company introduced biweekly project co-ordination meetings (covering all projects), supported by weekly status reports from project managers to the development manager. This improved project tracking, which again revealed some divergence between estimates and spent resources. The project manager was asked to cut off requirements, and one use case was dropped after discussion with the customer. The team also re-estimated the work of completing the necessary requirements. This estimate was approximately 120 working hours higher than the estimate at the end of the planning phase. It was taken a strategic company decision not to cut further functionality, due to the market potential of the product.

Apart from the left out use case and some detailing (mostly user interface issues) of one of the use cases, the developed system was quite according to the rather detailed requirement specification of the planning phase. At the end of the development phase, the requirement specification was updated according to these changes, and user documentation written based on the description of the use cases.

Because the software system was to be used as a demonstrator, the stabilising phase was rather brief, with only minor integration testing followed by some error correction. The system was found very stable for demonstration purposes. The baseline project ended with a demonstration and presentation of the developed software solution. The customer expressed great satisfaction with the delivered system. It is not unlikely that the customer will implement the system in a small scale "production system" – at least for demonstrational purposes.

During and after the baseline projects, results were measured and evaluated. In addition to continuously tracking actual work effort and progress, the more qualitative issues were found through meetings with the developers and project manager. Based on the results and evaluation, we have also come up with ideas for further improvements. The next section presents further details.

Measured Results and Lessons Learned

Below we describe some measured results and important lessons learned during the IMPOSE software process improvement project. Before concluding with some future actions, we also look closer at some lessons learnt in the perspective of a very small enterprise.

Measured Results

As the first baseline project was not completed, we could not measure actual effort and compare this to our estimates. Although we know that our competitor missed on cost estimates, the competitor's actual effort can not be correctly measured against our estimates.

However, combined with our detailed understanding of the problem domain and requirements and the detailed level of bottom-up estimation, this makes us even more confident that we avoided a significant economic loss and irresponsible commitment.

The second baseline project was completed through all phases, and we were able to compare the actual effort to our estimates. This is summarised in the figure below:

Overall

Phase	Estimate [h]	Used [h]	Deviation[h]	Deviation[%]
1	130	150.5	20.5	15.8
2	217	248.5	31.5	14.5
3	227	300.0	73.0	32.2
4	23	21.5	-1.5	-6.5
Totally	***597***	***720.5***	***123.5***	***20.7***

Fig. SAM.5: Estimates and actual effort of second baseline project

Half way into phase 3, the company's project tracking routines were changed, and this revealed deviation between actual and estimated costs. Counter measures were taken, which to some extent resulted in less functionality (in agreement with the customer). Still, this phase was overrun by 32.2 % compared to the final estimate of phase 2. The cause for this can to a large extent be linked to the implementation of two of the use cases, where new technology and development tools were used. More functionality could have been cut, but a strategic decision was made based on an evaluation of future market potential for the product. This resulted in a total overrun for the project by 20.7 %.

The average for a project of this size was 30.9 %, so our baseline is below that. The project was also delivered practically on time. But, although the quantitative goals were reached for both time- and cost overrun, we are not completely satisfied with the cost deviations. There certainly is room for further improvements.

Except for that, we are satisfied with the achieved software process improvements, which should be a solid foundation for further improvements. A qualitative analysis supports this view.

Some Lessons Learned

Through the work with the IMPOSE project, we have learnt several interesting lessons. Some of these are:

The more formal software process with tool support seems to have contributed to less uncertainty of what to do in the baseline projects. The activities are more standardised. In the baseline projects – and in particular in the second – far less resources than previous have been put into defining what to do and how to do it.

To us, use cases and object-oriented modelling are well suited to elicit user requirements. Also, use cases are also useful in structuring requirements as a basis for estimation and planning.

Standard deliverables through predefined templates are useful. Developers did not feel restricted by such templates, but found them to be useful tools in more efficiently producing and communicating quality documentation. In particular, this applied to SoDA templates for producing requirement documents based on the object-oriented modelling, but also for general documents and even MS Project templates.

In order to deliver a project within customer's constraints, it is important to know and understand these. It is important to early define a clear vision, but also a clear focus for the project, i.e. should features, resources (cost) or schedule be optimised? In order to succeed the customer, project manager, and the whole team should all agree on and follow these throughout the project. "Feature creep" or too high expectations are not necessarily customer-driven.

In both baseline projects, the developers felt that the most uncertain factor of the estimates was inexperience in the development tools (technology). This was the first times that project managers and developers felt higher risk concerning development tools than the software process.

The necessary skills for managing a software project should not be underestimated.

Also, the project leader should be followed-up regularly by resources external to the project.

Challenges related to SPI Projects in Small Companies

As a small company with only 14 employees (and only seven when planning the experiment), the work with the software process experiment has definitively been challenging.

The project has required key personnel of the company. However, key personnel are always wanted in other activities of a small company. This was particularly true at the start of the project, when the company went through a period of reorganisation. Although the SPI project was considered very important, it did not get enough backing and priority during the first months, to the extent it definitively deserved. This clearly improved after company reorganisation, when management could focus more actively on the SPI approach.

Although most employees have higher education in computer science, the company did not have much experience in performing software process improvements. As a very small enterprise, the company did not have resources dedicated to process improvements or software quality issues. When the IMPOSE project started, we had no experience in how to run such a project, and in general low competence in software process improvements. At project start, it might be a good idea to "walk" through the project in some detail, with some expert resources.

Another challenge has to do with the risk of planning baseline projects months in advance. Cancellations of development projects and new projects suddenly appearing implied rapid change of plans. We assume that this is more typical in smaller companies, and it sometimes makes the planning process difficult.

Improved SPI Maturity

The company has achieved quite a lot even if we are not completely satisfied with the measured results from the second baseline. An informal evaluation done by the authors showed that the second baseline project was able to meet the goals of CMMs key process areas: *Requirements management* and *Software project planning.* These goals can be formulated as follows (slightly rewritten):

software requirements are controlled to establish a requirements baseline (i.e. an analysed, stable set of documented requirements) for software engineering and management use;

software plans, products and activities are kept consistent with the software requirements;

software estimates are documented for use in planning and tracking the software project;

software project activities and commitments are planned and documented;

affected groups and individuals agree to their commitments related to the software project.

Even though this has not been of any major concern in the project, it is still interesting to notice. In order to achieve this, the following factors are critical:

application of a documented, iterative, use case driven and object-oriented software process, which clearly defines how to elicit user requirements, how to design the system, how to estimate the effort/cost and how to plan the project;
standardised and defined milestones and deliverables;
templates for important documents/deliverables, such as the vision/scope document, project structure document, the requirement specification document, and the project plan, all of which will be reused in later projects;
high degree of tool support, including tools for object-oriented modelling (with use case modelling), tools for generating the requirement specifications and tools for requirements management, analysis, and evaluation (estimation).

From an organisational perspective, we should also mention:

higher knowledge of software process improvements in general, both for the management and the developers of the company;

acknowledgement of and enthusiasm for the software process improvements among developers and the management of the company;

higher knowledge and experience in eliciting and managing user requirements, and in object-oriented modelling using the Unified Modelling Language;

higher competence among developers in using supporting software tools.

A small company has some interesting advantages for software process improvements. With 7-14 employees during the IMPOSE project period, it has been relatively easy to involve the whole organisation. Several meetings have been hold where improvements and results have been presented and discussed. On these occasions valuable feedback have been given for further improvements.

Future Actions

It is not likely that Invenia will start a software process improvement project of this size in the near future. This is *not* because the challenges of the IMPOSE project has scared us from such projects, but because we believe that smaller adjustments will now give better cost/benefit. In particular, we will look closer at issues like:

routines and tools (simple Excel spreadsheet or check list / matrix) to ensure that the customer and team early agree upon whether to optimise features, resources (cost) or schedule;
a simple project manager’s check list for each of the four phases, to ensure that important activities are carried out;
further improvements for project management, including simple tool support (like TimeSheet reports and Excel templates) for regularly (weekly) structuring and presenting up-to-date actual effort and remaining work for all activities and (use

cases) of the current phase;

defining more and improving existing templates for “reusable deliverables”, in order to speed up the activities of particularly the first two phases;

upgrading the supporting software tools and adjust the use of these to improve the integration between these tools (object-oriented modelling, document automation, and requirements management).

It is also critical that the software process model is being used in future projects, and that deviations are discovered and analysed. One of the important tasks of the company’s development director is to ensure that the process is followed when possible and analyse deviations in order to assure that possible improvements are discovered and handled in a controlled manner.

On a longer term, we may define additional software process improvement projects. Currently, we have defined the following areas as interesting for further process improvements:

a closer adoption to the *team model* of the Microsoft Solution Framework;

reuse, including software components/source code as well as conceptual, logical, and physical design (i.e. reuse of use cases, classes and code);

testing and verification;

configuration management.

Acknowledgement

The IMPOSE project is a Process Improvement Experiment (PIE) sponsored by the CEC under the ESSI Programme, project no. 23780. The financial support for the project is gratefully acknowledged.

We also want to acknowledge Peter Holmes and Jon Ølnes, Norwgian Computing Center, and Lars Kristoffersen, Invenia AS for reviewing this paper.

References

[1] Gause D.C., Weinberg G.M., *Exploring Requirements: Quality berfore Design*, Dorset House Publishing Company, 1989.

[2] Sommerville I., P.Sawyer P., *Requirements Engineering: A good practical guide*, J.Wiley & Sons, 1997.

[3] Paulk M.C., Weber C. V., Curtis B., M.B.Chrissis (ed.), *The Capability Maturity Model – Guidelines for improving the software process*, Addison Wesley Publishing Company, 1995.

[4] Stone J.A., *Developing Software Applications in a Changing IT Environment*, McGraw-Hill, 1997.

[5] Jacobson I., Christerson M., et al*., Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992

[6] Fowler M., Scott K., *UML Distilled – Applying the Standard Object Modeling Language*; Addison-Wesley, 1997.

[7] McConnell S., *Rapid Development: Taming Wild Software Schedules*, Microsoft Press, 1996.

[8] Quatrani T., *Visual Modeling with Rational Rose and UML*, Addison-Wesley, 1998.

[9] Rational Software, et al, *Unified Modeling Language: UML Summary, version 1.1*; Rational Software, 1997 (document available on the World Wide Web at www.rational.com/uml.)

[10] Rational Software, et al, *Unified Modeling Language: UML Notation Guide, version 1.1*; Rational Software, 1997 (document available on the World Wide Web at www.rational.com/uml.)

[11] Microsoft Education and Certification, *Solutions Development Discipline*; Course No.493; Microsoft Corporation, 1997. (More information is available at http://www.microsoft.com/solutionsframework)

[12] IMPOSE, *Mid-term report,* 1998.

[13] IMPOSE, *Final report,* to be published.

Appendix A: CV for the Authors

Mr. Svein A. Martinsen

At the age of 29 years, Mr. Martinsen is currently working as the Development Director of *Invenia AS*. He is responsible for the software process improvements of the company, including the work with IMPOSE.

Svein has a M.Sc. degree in software engineering from the University of Science and Technology, Trondheim, Norway (1993). In 1993-1994 he did military service, where he attended a sergeant course and worked at the IT section of Bardufoss air station. During 1994-1995 he continued working for the Norwegian Air Force, mainly with development and implementation of Geographical Information Systems.

In 1995 he joined DataKompetanse AS (now Invenia AS) where he worked as a software developer until October 1997, when he joined the company management as the Development Director. He has participated in several client/server development projects, both as a developer and as a project manager. He has attended several courses like *Requirements Management* (1998), *Microsoft Solution Framework* (1997), *Object-oriented Analysis and Design with the Unified Modeling Language Using Rational Rose* (1997), *Risk Management of IT-projects* (1997), *Software testing* (1996), and *Product development* (1995).

Mr. Arne-Kristian Groven

He is working as a research scientist at the Norwegian Computing Center (NR) in

Oslo. In the IMPOSE project he has been involved as a software process improvement consultant in some of the work packages of the project.

Arne-Kristian has a M.Sc. degree from the Institute for Informatics, University of Oslo, where he also was employed as a research assistant from 1991 to 1992. Between 1992 and 1996 he has been working with formal analysis, -specification and -design methods at the Institute for Energy Technology, OECD Halden Reactor Project, in Halden, Norway.

Since 1996 he has been working at the Norwegian Computing Center, exclusively with software process improvement activities. He has been involved in several SPI projects, both within ESSI and national SPI programs. In these SPI projects he has been assisting several Norwegian software development companies.

Appendix B: Company descriptions

Invenia AS

Invenia AS is a small software development organisation located in the northern part of Norway. The company has 14 employees, of which 10 are producing tailor-made software (client/server) for customers, often on a fixed-price basis. The business idea of the company is to help customers realise their business goals through strategic IT-concultancy and development of software solutions supporting the business processes of the customer’s value chain.

Invenia AS specialises in client/server and collaboration solutions for Windows, Windows NT, and the Internet. Development tools include MS Visual Studio (Visual Interdev, Visual Basic), MS FrontPage, C++, and PowerBuilder. As a server platform we focus on Windows NT with Internet Information Server, MS Exchange, and relational databases such as MS SQL Server, Sybase SQL Server, and Sybase SQL Anywhere.

All developers are educated through university studies or college, and have work experience ranging from 0 to 8 years. The general manager is Mr. Terje Wold, which also is the owner of the company. Customers come from both the private and public sector, some of which are large national companies. However, our market focus is in Northern Norway. During the first half of 1998, Invenia AS (VAT reg. no.: 960930401 MVA) had an operating income of approximately 4,0 million NOK, and an operating result of approximately 1,6 million NOK.

Norwegian Computing Center

The Norwegian Computing Center (NR) employs a staff of 90 who are engaged in projects within the fields of information technology, statistical and mathematical modelling. The main areas include security, object-orientation, interactive media, electronic commerce and EDI, statistical data analysis, video analysis, remote sensing, and business development with IT.

Our main customers are within Norwegian business and public administration. NR also takes part in national and European research programmes. The author from NR is working within a group focusing on research and consultancy within the area of object-orientation, security, and software process improvement.

Session 8 – Information and Team Management Solutions for SPI

NQA - Experience with Integrated Teamwork and Network based Quality Assurance

Dr Richard Messnarz, Director ISCN Ltd., Dublin, Ireland
Dr Robert Stubenrauch, Joanneum Research, Graz, Austria
Martin Melcher, Rainer Bernhard, NQA Chief Developers, ISCN, Graz, Austria

Promotion of an ISO9001-based quality system using the WWW in a software organisation and its experience

Atsuo HAZEYAMA, Katsumi HONDA
Client-server Software Laboratories,
NEC Corporation, Tokyo, Japan

The PASS Software Process Improvement Experiment in Hungary (PASS EP 21223)

János Ivanyos
MemoLuX Ltd., Hungary
Dr. Miklós Biró
MTA SZTAKI, Hungary

NQA - Experience with Integrated Teamwork and Network based Quality Assurance

Dr Richard Messnarz, Director ISCN Ltd., Dublin, Ireland
Dr Robert Stubenrauch, Joanneum Research, Graz, Austria
Martin Melcher, Rainer Bernhard, NQA Chief Developers, ISCN, Graz, Austria

Summary:
This paper describes the experience with a new team management concept for software quality assurance which has been field tested in firms since 1994 in different countries and resulted in an IEEE book "Better Software Practice for Business Benefit".

In addition, since 1998, the European Commission supports the development of a tooling concept on basis of Hyperwave (a leading multimedia system which won the EU IT Prize in 1997) for computer automating these role centred team work management paradigms for quality management. At the time of paper writing it is field tested in leading German and Scandinavian firms and research centres such as Daimler Benz, FZI, etc.

This paper contains a description of the paradigms followed and a first summary of the lessons learned.

Richard Messnarz, Dr., Florence House, 1 Florence Villas, Bray, Co. Wicklow, Ireland, ph.: +353 1 286 1583, fax.: +353 1 286 5078, email: rmess@iscn.ie

Introduction

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 (e.g. selling the product then in more regions of Europe through VARs - Value Added Re-sellers).

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.

NQA is developed in co-operation with Hyperwave [6], [7], a leading information management system, and places the required functionality of such a virtual office on top of it. The system has been field tested in leading German and Scandinavian industry in the software co-operation and out-sourcing sector.
Most of the large companies (e.g. Siemens re-organises again and centralises more, Daimler Benz and other manufacturers require a much more closer co-operation between the teams of suppliers and their internal teams) start to question the simple out-sourcing concept. They think about a system that will create a virtual office through which teams from the supplier and the manufacturer work together as if they are one team in the same office, thus solving the missing-control problem of out-sourcing.
Another potential is to use it as a virtual office of many smaller companies who jointly develop and follow joint quality control scenarios over the net.

The development of NQA has been supported by the European Commission under the ESPRIT project HYMN.

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 [8]) and about which there is a book published by IEEE at the end of 1998 [10]: Better Software Practice for Business Benefit - Principles and Experience (ed. Richard Messnarz, ISCN).

The Underlying Management Principle

A software process is not seen as just a sequence of tasks with a planned result [1], but it is the result of an integrated team work environment [10]. 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 [9], [10]

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 [2], [3], [4], [5].

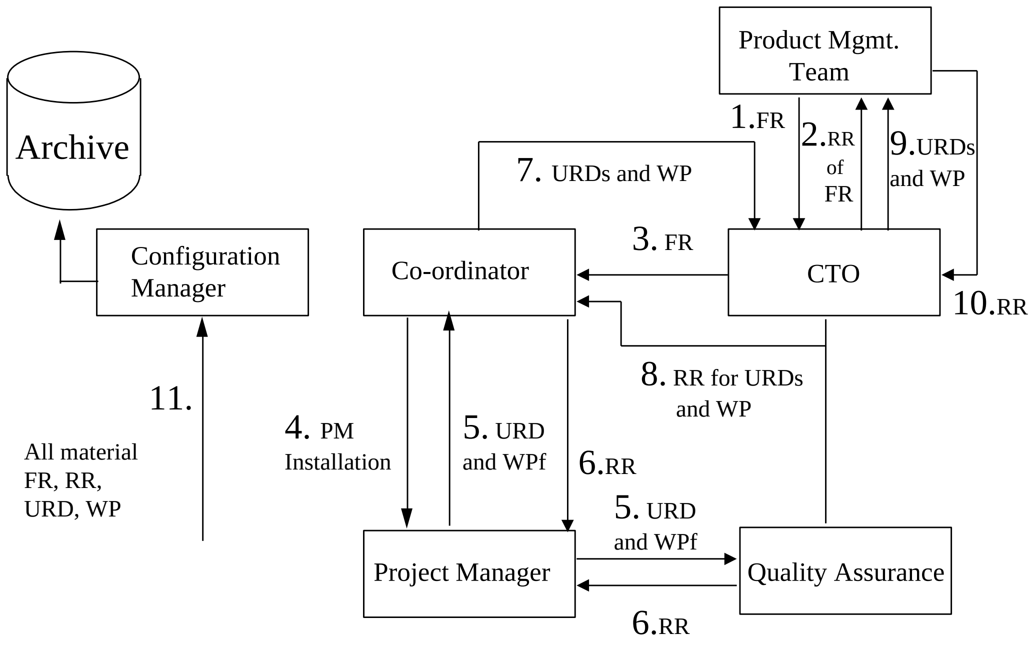


Figure 1: A Role Play for Feature Request Management and Planning

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).

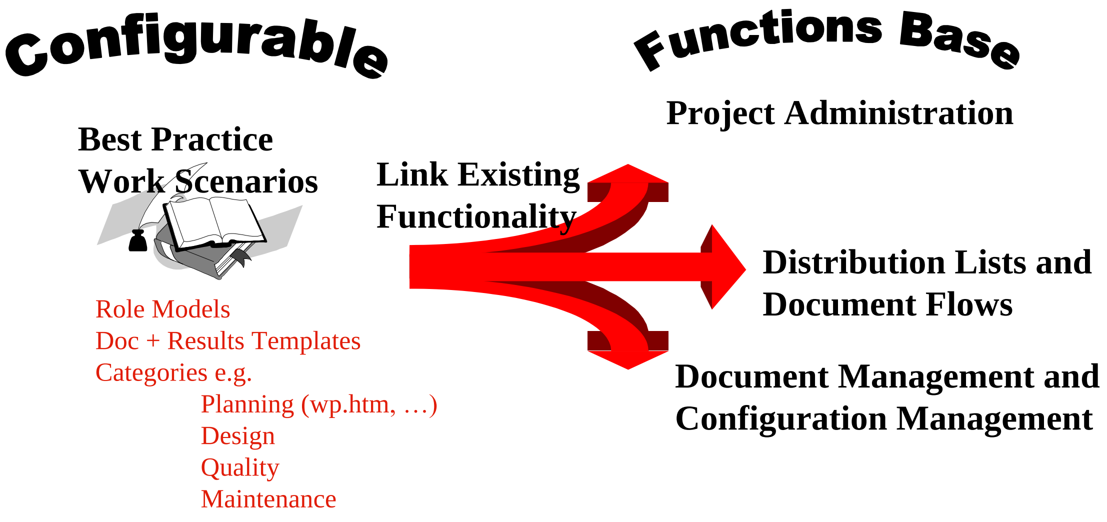
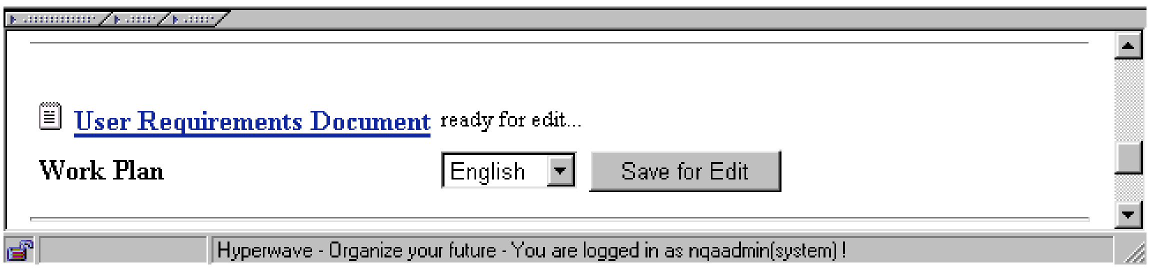


Figure 2: Data and Functional Configurability

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)



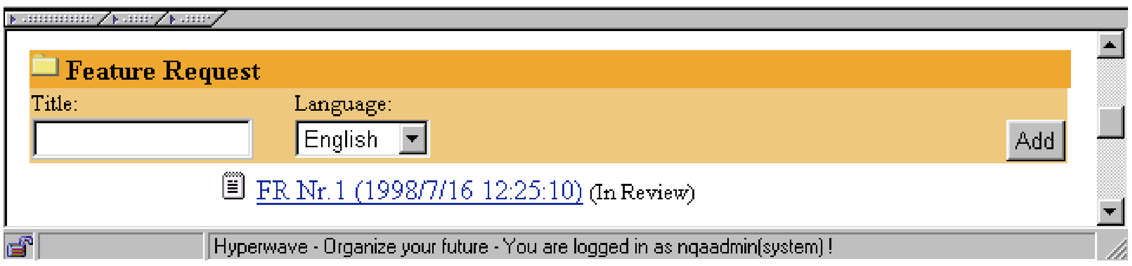
User Requirements Document ready for edit...

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:

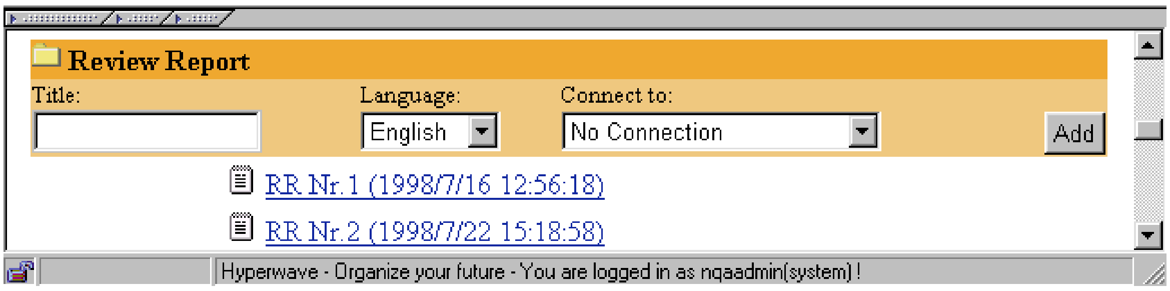
Language: English

FR Nr.1 (1998/7/16 12:25:10) (In Review)

Add
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

RR Nr.1 (1998/7/16 12:56:18)

RR Nr.2 (1998/7/22 15:18:58)

Add
Hyperwave - Organize your future - You are logged in as nqaadmin(system) !

Figure 5 : Linked Report Object Creation Window

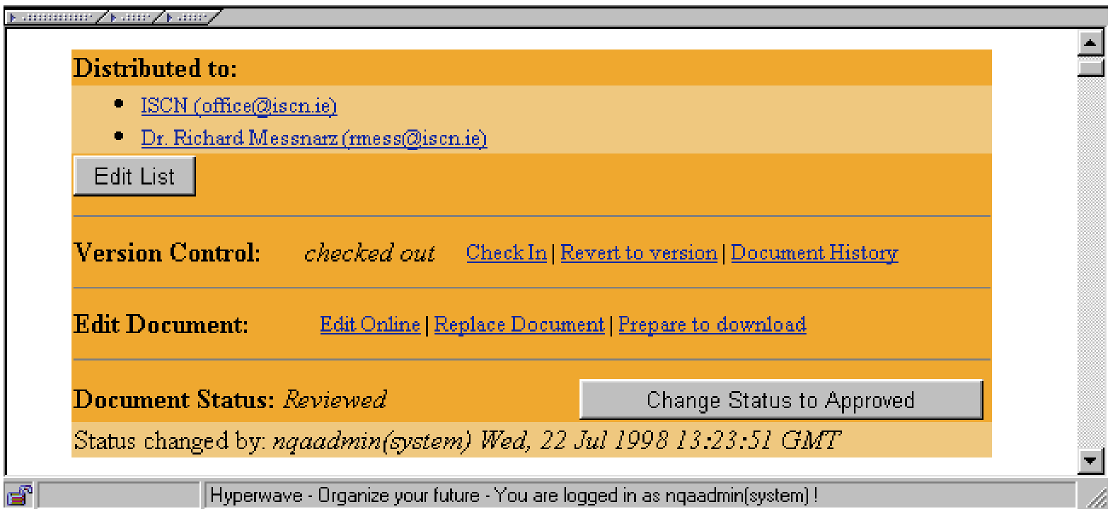
Depending on the user needs the three elements are configured. E.g. Linking Feature Requests (FR) with user Requirements Documents (URD), so that an URD is automatically created by the links to accepted FRs (example from a customer wish from Daimler Benz).

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.



Distributed to:

- ISCN (office@iscn.ie)
- Dr. Richard Messnarz (rmess@iscn.ie)

Edit List

Version Control: *checked out* Check In | Revert to version | Document History

Edit Document: Edit Online | Replace Document | Prepare to download

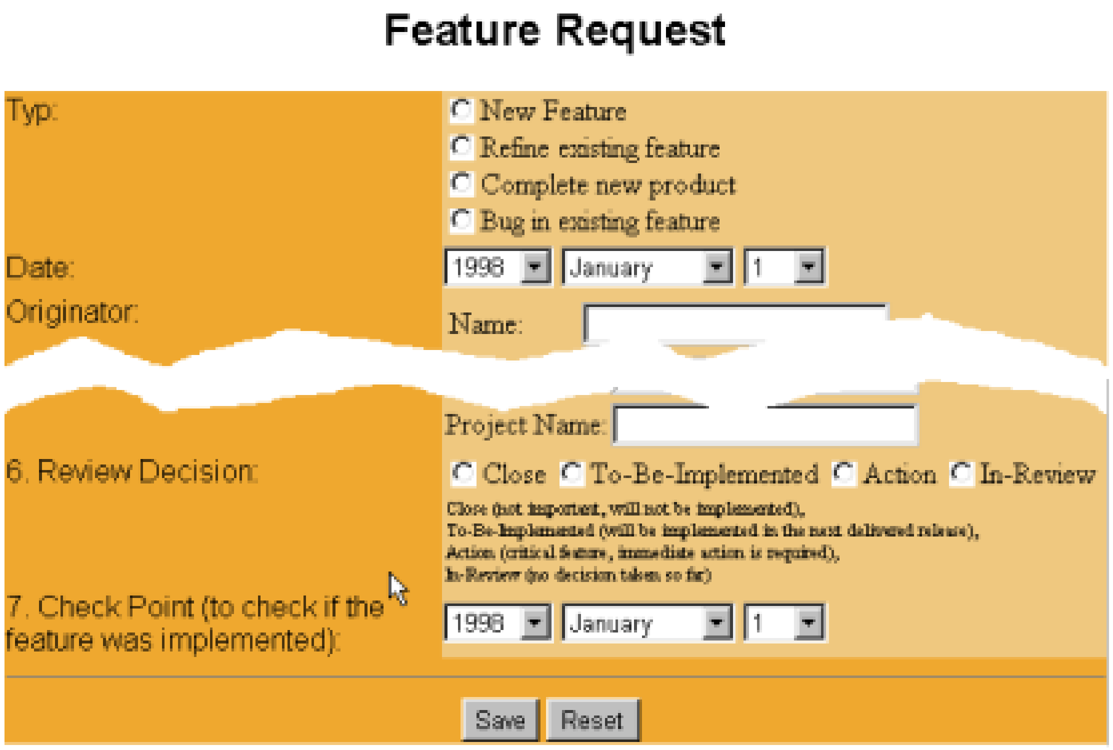
Document Status: *Reviewed* Change Status to Approved

Status changed by: *nqaadmin(system) Wed, 22 Jul 1998 13:23:51 GMT*

Hyperwave - Organize your future - You are logged in as nqaadmin(system) !

Figure 6: Functional Header Generated for Document Objects

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).

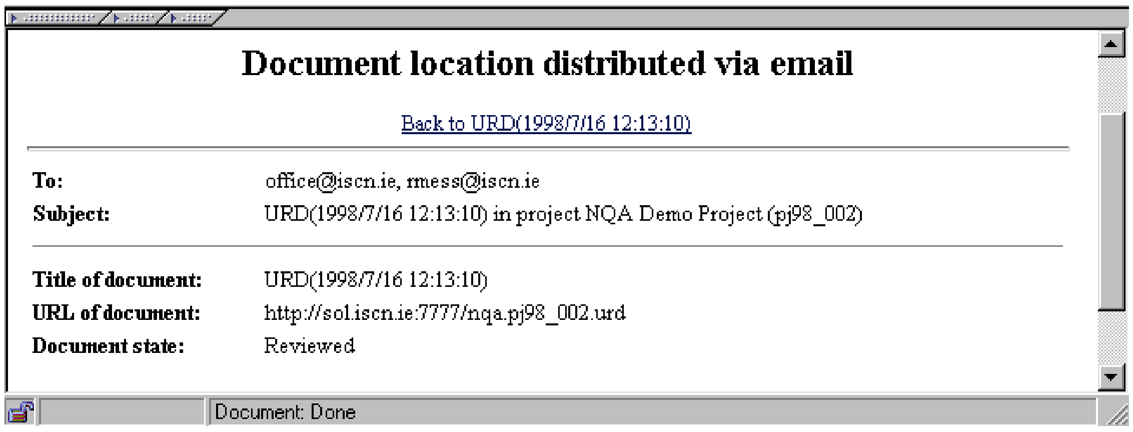


Feature Request

Typ:	New Feature Refine existing feature Complete new product Bug in existing feature
Date:	1998 January 1
Originator:	Name:
	Project Name:
6. Review Decision:	Close To-Be-Implemented Action In-Review Close (not important, will not be implemented), To-Be-Implemented (will be implemented in the next delivered release), Action (critical feature, immediate action is required), In-Review (no decision taken so far)
7. Check Point (to check if the feature was implemented):	1998 January 1

Figure 7: Parts of a Typical Report Form

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).



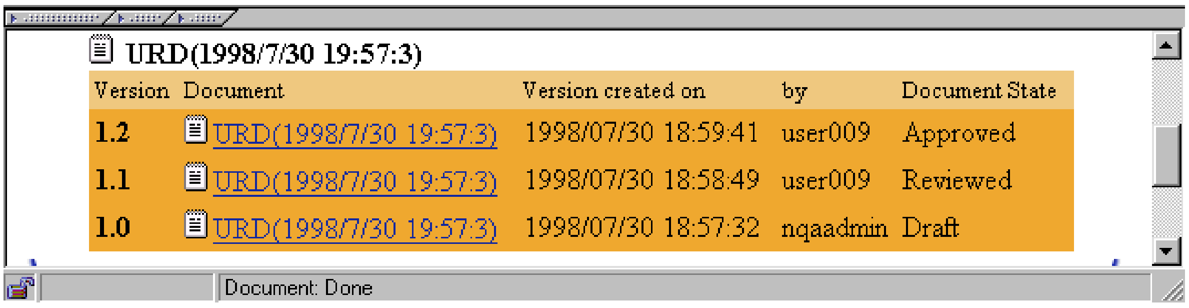
Document location distributed via email

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 8: 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 9: 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.

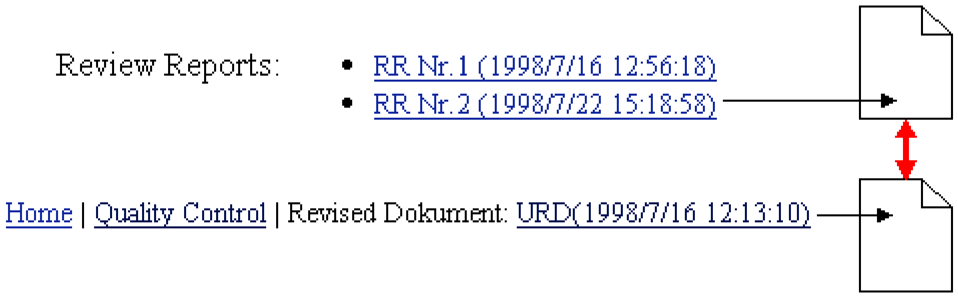
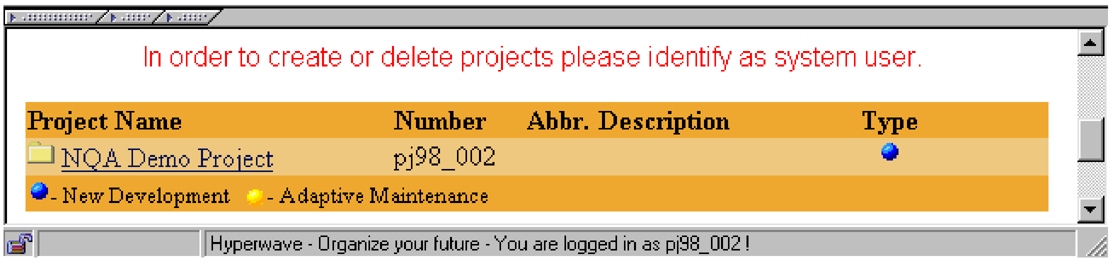

Review Reports:
• RR Nr. 1 (1998/7/16 12:56:18)
• RR Nr. 2 (1998/7/22 15:18:58)
Home | Quality Control | Revised Dokument: URD(1998/7/16 12:13:10)

Figure 10: 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.



In order to create or delete projects please identify as system user.

Project Name	Number	Abbr.	Description	Type
NQA Demo Project	pj98_002			●

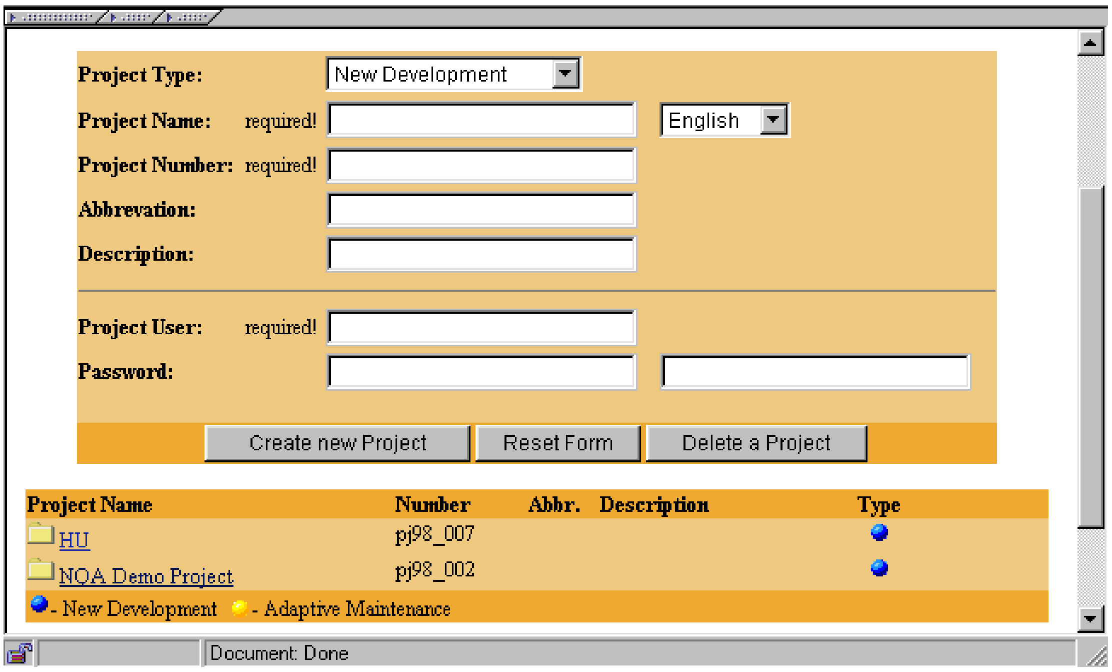
● - New Development ● - Adaptive Maintenance

Hyperwave - Organize your future - You are logged in as pj98_002 !

Figure 11: 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.



Project Type:		New Development	
Project Name:	required!		English
Project Number:	required!		
Abbrevation:			
Description:			
Project User:	required!		
Password:			

Project Name	**Number**	**Abbr.**	**Description**	**Type**
HU	pj98_007			●
NQA Demo Project	pj98_002			●

● - New Development ● - Adaptive Maintenance

Document: Done

Figure 12: Creation of Projects also Builds Project Teams

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.

Experience with NQA Concepts in Industry

Some case sstudies from real practice are -

Case Study 1 - System Use

This relates to an experience from a supplier and large manufacturer relationship. All Feature Requests (FRs) to know the requirements of a system are jointly collected over time, and accepted ones result in a User Requirements Document (URD).
This process has been extremely difficult because the teams were locally distributed resulting in instable URDs with changes and additions lateron.
A stability of approx. only 50% was given, and a high flexibility in the entire process was needed. A much better situation is the goal with about 80% stability and fewer additions later, leading to higher effort at the start but much higher productivity throughout the development and field test.
Now, a virtual office system allows to jointly collect the FRs, to publish them in a joint planning cluster, to submit them for review over the net, and to automaatically link them forward and backward with the related reviews leading to either acceptance or non-acceptance of the FR.

The most important feature was then the automatic forward and backward linking of the accepted FRs with the URD, leading to an automatically created URD on the net.

Case Study 2 - Team Management in SMEs

It has to be noted that the role centred team management as a principle works even without a tool support. It has been tested in SMEs in different countries.

A case study from a Spanish innovation management organisation showed that the use of role based team scenarios is a perfect tool for training of new staff to be integrated. The roles are clear from the beginning and to know the interfaces to other team members in advance led to much easier integration of new staff. The average time to integrate new staff was reduced by 67%.

A case study at a middle sized software and research company in Austria showed that the use of real practice work scenarios for the establishment of working instructions for an ISO certification leads to a large increase of the motivation of staff. (17% really using to above 70%).

Another factor is that practitioners want to have a system in place (not a paper handbook) that facilitates them the additionally required documentation effort and is accessible through a common interface from their normal work place.

Pitfalls

This new concept did not work -

If organisations tried to use it without understanding the required management principles first.
When middle management realised that the role models also require that their role is described with interfaces to the roles played by the staff, because this also makes their duties more visible. (a psychologic barrier)
If the system 's configurability and linking features were neglected or not properly configured leading to inflexible top-down project administration (hated by technical staff).

Acknowledgements

We are grateful to the European Commission for supporting us in the development of this new working concept. The research work and the underlying IEEE book have been supported under EU Leonardo da Vinci Programme in the PICO project.
The EU ESPRIT Programme supported us to realise the ideas and new comcepts in an industrial tooling environment on the basis of Hyperwave in the HYMN ESPRIT project.
We would like to thank Gerhard Pail, Frank Kappe, Mansuet Gaisbauer, Juergen Schipflinger, Joerg Faschingbauer, and Peter Luttenberger for their active support, the test, and the use of NQA for ISO 9001 certification of Hyperwave itself.

We also would like to thank Mr Alejandro Moya from the European Commission for his strict control to ensure project progress, and all people from Hyperwave, Daimler Benz, FZI who field tested the system.

References

[1] 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

[2] 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).

[3] ISO 9001. Quality Systems. Model for Quality Assurance in Design/Development, Production, Installation and Servicing. International Organisation for Standardisation, Geneva (1987)

[4] ISO 9126, Information Technology - *Software Product Evaluation* - Quality Characteristics and Guidelines for Their Use, 1991

[5] ISO/IEC 12207, *Information technology - Software life cycle processes*, first edition Aug. 95.

[6] Maurer H., Necessary Ingredients of Integrated Network Based Learning Environments; *Proc. ED-MEDIA'97*, Calgary, AACE (1997), 709-716.

[7] Maurer H., What We Want from WWW as Distributed Multimedia System; *Proc. VSMM'97*, Geneva. IEEE, 148-155.

[8] Messnarz R., Mac an Airchinnigh M., Biro M., Tully C., ISCN - An International Software Collaborative Network, in: *Proceedings of the ISCN´96 Conference on Practical Improvement of Software Processes and Products in December 1996 in Brighton/London,* ISCN Ltd. Dublin, Ireland, 1996

[9] 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

[10] Messnarz R. , Tully C. (eds.), The PICO - Book *: Software Process Improvement for Business Benefit - Principles and Experience*, IEEE Computer Society Press, November 1998

Promotion of an ISO9001-based quality system using the WWW in a software organisation and its experience

Atsuo HAZEYAMA
Client-server Software Laboratories,
NEC Corporation, Tokyo, Japan

Katsumi HONDA
Software Design Laboratories,
NEC Corporation, Tokyo, Japan

Introduction

In recent years, software industry have had to overcome global competition in markets as well as other industries like automobile, semiconductor, home electronics, computer hardware, etc. Companies/organisations in this industry are required to provide attractive and high-quality software products on time and within budgetary limits while continuously improving their processes [1].

NEC Corporation not only is a manufacturer of communications, computers, and electron devices but also deals with software business. Software business plays a very important role in NEC in recent years; Around 10 thousands of system engineers (at the end of March 1998) belong to the C&C[1] system business unit where provides total solutions to wide varieties of customers by the system integration which combine hardware with software. The non-hardware sales of computer business in NEC in fiscal ending March 31, 1998 was around 565 billion yen (around US$ 39 million) (increased 13 percent from the previous fiscal year). The ratio of non-hardware business (including SI business, program products sales, maintenance, etc.) of all the computer business becomes 31.6% (increased 2.3 percent from the previous fiscal year) at the end of 1998 fiscal year. The importance of software process improvement, therefore, has been recognised.

Based on the background, NEC developed a methodology of software process innovation. It is consisted of three steps. The first step of the framework is to build an improvement basis based on the ISO9001 quality system. This step is therefore important for effective and efficient improvement, especially with respect to establishing processes based on the ISO9001 requirements, and managing documents and quality records. This paper describes a methodology of software process innovation in NEC and a case study of a software development organisation in NEC for promotion of an ISO9001-based quality system, using the Web computing especially documents and quality records management and their sharing.

[1] trademark of NEC Corporation.

A framework of software process innovation in NEC

This section describes an overview of a framework of software process innovation in NEC. The detail of the framework is written in [2].

NEC has applied Total Quality Control (TQC) activities to software divisions since 1981[4][5] (we call these activities Software Quality Control (SWQC)) and some achievements have obtained. However, as these activities were carried out independently in each business division, the results varied depending on the improvement approach of each division.

In the last decade, several software process improvement methods such as CMM (Capability Maturity Model) [6], ISO9001 [3] were proposed.

NEC considers it possible to make effective process improvement by regarding the above three approaches as having different characteristics respectively and organically associating their characteristics.
The steps toward process innovation by using the ISO9001-based quality system, the process maturity model, and the QC techniques are shown as follows.

STEP1: ISO9001 Quality System Construction Stage
Most of software divisions has not yet established software quality system. The first thing for such an division to do for the process innovation is therefore to build an improvement basis. ISO9001 is used for constructing this improvement basis. The following activities are required:
to **define the process by documentation and thoroughness of its implementation**
to promote improvement activities through periodical audits
to provide a good goal for construction of this quality system, which is ISO9001 certification.

A case study of a software division in this stage is shown in the next section.

STEP2: Improvement Stage
A division which has established the ISO9001 quality system stands in the second stage in which well-planned improvement is continuously carried out along the improvement management cycle, Plan-Do-Check-Action (PDCA) Cycle.
The improvement management cycle is used to comprehend the current situation, to extract problems, to decide the priority levels, to make well-planned improvements, and to evaluate the results.
In implementing this improvement management cycle, the ISO9001 quality system, the process maturity model and the QC techniques are utilised. This ensures real results including productivity and quality. The improvement activities are carried out based on analysis not only of problems discovered but also of examples of success.

STEP3: Business Effects Evaluation and Accumulation Stage
The final stage is to evaluate the improvements accumulated in the improvement management cycle and to check their contribution to business. This is used to verify the validity of the approach to improvement and gather information on the effects. A good goal will be to win prizes such as the MB Award of the United States and the Japan Quality Award, which commended the improvement having contributed to businesses and played a huge role.

A case study of promotion of an ISO9001-based quality system using electronic media in a software development organisation in NEC

This section describes a case study of promotion of an ISO9001-based quality system using electronic media in a software development organisation in NEC. The organisation in this case study develops software products that support software development divisions in our corporation. The top management of the organisation decided to establish an ISO9001-based quality system to improve the quality of its own processes and products.

The organisation has several relatively large projects. Projects in the organisation are traditionally operated by using electronic media and computer network (E-mail, the WWW, etc.). Most artefacts are created using computers, and various kinds of communication (reporting, notification, discussion, review and inspection, document distribution, etc.) are done using E-mail. Such characteristics were taken into account in constructing the ISO9001-based quality system.

The characteristics of the ISO9001-based quality system with respect to document control and information sharing of the organisation

When software development is performed in the form of projects which are consisted of some people, some rules which should be observed by project members are required. The larger the size of projects, the more the importance such rules are.

Before the ISO9001-based quality system was introduced in the organisation, each project established some procedures for its own project: procedures such as rules for naming variables and constants in source codes, rules for document control, and guidance for manual writing. A document control rule of a large software project which specified document identification number and the storage directory, was adopted as the basis of the document control standard in the organisation's quality system. But because the document control rule of the project did not explicitly define what documents should be created and managed, the decisions about which documents were to be recorded was left to each of the project / sub-projects. As a result, some important documents were not recorded (they remained in E-mail spools or they were stored only on papers). Some documents were even lost because their authors moved. As a result, problems have occurred that design rationales could not be found out.

In building the ISO9001-based quality system, a document control standard was established which specified the process from document creation, inspection, approval, registration, distribution, till invalidation, document structure, document identification number, the storage directories, as well as a document and quality record list to be created and maintained through the quality system. Here quality records are things which are required to maintain by the ISO9001 standard `4.16 Control of quality records'. On the other hand, documents are things except the quality records of all the things which are required to maintain by the ISO9001 standard.

This list contains around 50 documents and quality records, and for each the following attributes are specified: creator, inspector, approver, distribution, whether or not it is a quality record, the directory within which it is stored, how long it should be stored, the name of the standard specifying the details about the document and quality record, and whether or not there exists a template for the document and quality record (one example is shown in Table HAZEYAMA1).

Table HAZEYAMA 1: Example attributes and their value of a document Controlled under the quality system (development planning document)

Attribute	Attribute value
Creator	persons who are appointed by project leader
Inspector	project leader
Approver	top management
Distribution	the whole project
Quality record or not	N/A
Storage directory	project directory
Storage duration	five years
Standard name	planning standard
Existence of template	existence

Since the ISO9001-based quality system was applying, the amount of documents and quality records recorded by projects/sub-projects increased rapidly. Table HAZEYAMA2 shows the changes in the number of documents and quality records created and stored during the development of a large project within the organisation.

Table HAZEYAMA 2: The changes in the number of documents and quality Records created and stored within a large software project

Product version No.	No. of documents/ quality records	Characteristics of the development
V1.0	190	First release
V2.0	77	Minor version-up
V3.0	79	Porting to another platform and bug fix
V4.0	1022	Major version-up. Start applying the quality system to the organisation. 1st survey was performed.
V4.1	461	Some enhancement and bug fix.
V5.0	821	Major version-up. Received the examination. 2nd survey was performed.

This table shows that the introduction of the ISO9001-based quality system (at the time of V4.0 development in this table. 1022 documents and quality records were created in this development.) resulted in a five-fold increase in the number of recorded documents and quality records over the numbers recorded when the first version was developed (190 documents and quality records were created in V1.0 development.)

This increase in the number of documents and quality records resulted in increased retrieval cost, too, therefore to systematize information sharing was required. Considering this background, we proposed an information sharing system called electronic binder system using the WWW and deployed it in the organisation so that people can share information for software development and can prepare the internal quality audits and certifications effectively and efficiently.

Overview of the electronic binder system

The electronic binder uses the WWW and HTML (Hyper Text Mark-up Language) to collect computer-created documents and quality records according to a classification scheme specified later. It differs from a paper binder in that a lot of binders can quite easily be constructed from various viewpoints without replicating documents and quality records themselves.
Fig. HAZEYAMA1 shows the configuration of the electronic binder system. It is a very simple configuration in that persons register documents and quality records in a centre file directory and responsible persons create and maintain an electronic binder by using a template. We can view the electronic binders by using a Web browser.
The electronic binders provide various viewpoints because a lot of entries in a lot of electronic binders (HTML files) can link to a single document and quality record regardless of where they are registered.
It is particularly useful in a large software project because such a project is composed of several sub-projects which have their own sub-project document directory, and the electronic binder of the whole project can be created by aggregating the binders of its sub-projects. The electronic binder of each product version can also be created very easily. Furthermore the construction cost of the electronic binders is very low because of the use of the WWW.

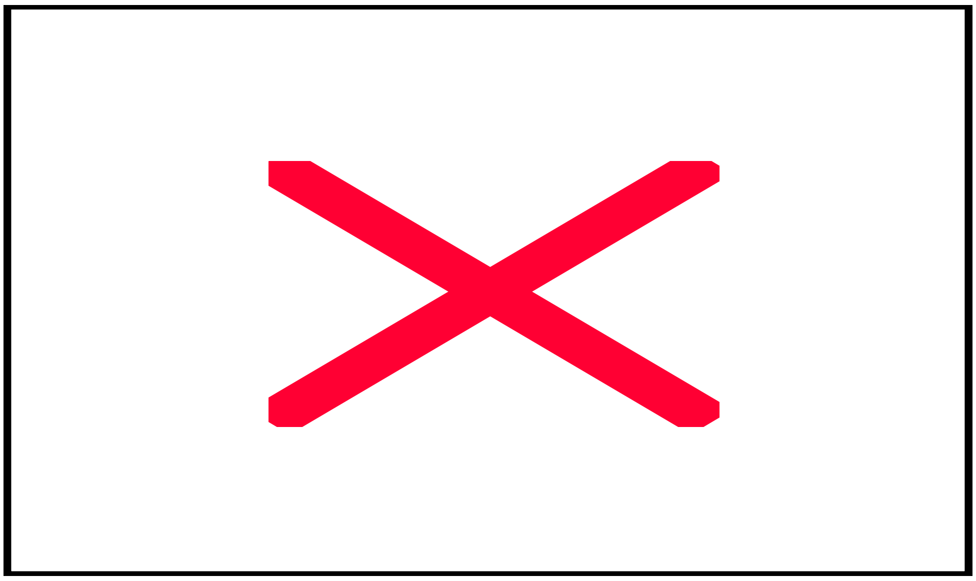


Fig. HAZEYAMA 1 : Configuration of the Electronic Binder System

Fig. HAZEYAMA1 shows the configuration of the electronic binder of a large project which is composed of several sub-projects. This figure shows that the binder of the overall project is composed of documents and quality records which are created for the overall project, and the binders of its sub-projects. This figure also shows the binders for multiple product version of the project (version N development and version N+1 development). Using the electronic binder system, documents which are shared among several product version such as file-a, file-c, file-e in this figure can be easily accessed from the binder for each product version. Fig.HAZEYAMA2 shows a screen image of the electronic binder of a project.

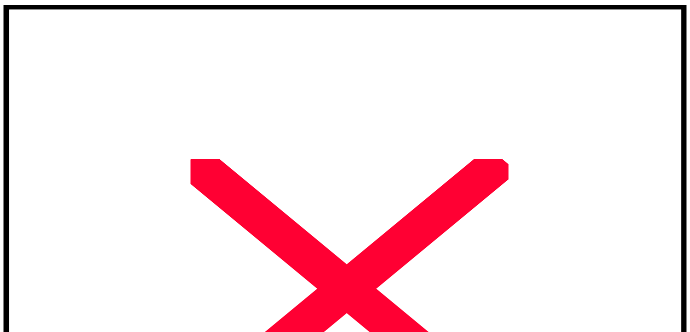


Fig. HAZEYAMA 2 : Screen Image of an Electronic Binder

Deployment of the electronic binder system

The electronic binder system has been deployed throughout the whole organisation in the following way:
(1) Documents and quality records to be managed according to ISO9001 requirements were extracted and defined by an ISO9001 promotion team (The list of these documents and quality records forms a part of the quality system). These documents and quality records were divided into two categories:
Documents and quality records created and maintained by the organisation.
Documents: contract documents, internal quality audit plans, training plans, etc.
Quality records: internal quality audit reports, each personnel's training records, and management review reports, etc.
Documents and quality records created and maintained by the projects and/or sub-projects.).
Documents: development plans, design specifications, test specifications, production specifications, final inspection and testing specifications, etc.
Quality records: test reports, various review reports, final inspection and testing reports, corrective / preventive action reports, etc.

Templates of the electronic binder for the organisation and for the projects / sub-projects were prepared by the ISO9001 promotion team.

(3) A procedure for the application of the electronic binder system was documented by the ISO9001 promotion team and approved by the top management. This procedure defined who should create and maintain each electronic binder and where each electronic binder should be placed. It also defined the file name conventions of electronic binders. This document was distributed to the whole organisation. The organisation and projects / sub-projects are required to create and maintain their own electronic binders by using corresponding templates.

Survey of the electronic binder usage

As we surveyed the usefulness of the electronic binder system by circulating questionnaires to the whole organisation, we show the contents of the survey and the results.

Status at the time when the survey was performed

The 1st survey was performed October 1996, soon after having introduced the electronic binder system to the organisation. When this survey was performed, the status of the quality system was as follows: a large project in the organisation had been applying the quality system for the second time (Version 4.1 in Table HAZEYAMA 2). Around 65 % of the organisation have been participating in this project with part-time or full-time. As Table HAZEYAMA 2 shows, V4.1 development was a minor enhancement and bug fix. On the other hand, as activities for the organisation, planning and training record of the quality system, and each personnel's training record had been created and maintained.
When the survey was performed, internal quality audits, and management reviews had not been performed yet.

The organisation received the ISO9001 formal examination July 1997 and received the certification September 1997. The 2nd survey was performed September 1997, soon after having received the certification.

Contents of the survey

We asked all the employees of the organisation, and employees of subsidiary companies associated with projects of the organisation for the following information:
(1) their affiliation
(2) whether they had accessed the electronic binders
(3) the frequency with which they had accessed the electronic binders
(4) which electronic binders they had accessed
(5) what documents and quality records in the electronic binders they accessed
(6) how useful they had found the electronic binders
(7) reasons that the electronic binders are useful
(8) reasons that the electronic binders are not useful
(9) key success factors of software process improvement via ISO9001

(9) is an item which was asked only in the second survey.

Results of the survey

In the 1st survey, sixty-three percent responded and seventy-three percent of those responding had referred to the electronic binders. In the 2nd survey, fifty-six percent responded and eighty-three percent of those responding had referred to the electronic binders. The following results came from the data which people who had accessed to the electronic binder responded.
The results on evaluating the usefulness of the electronic binder are shown in Fig. HAZEYAMA 3.

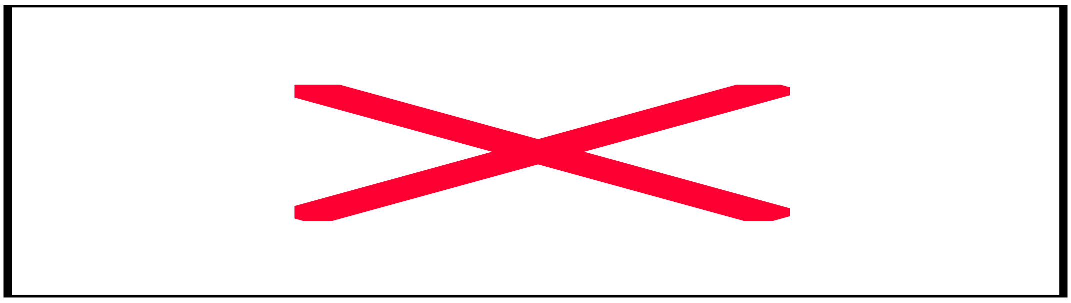


Fig. HAZEYAMA 3: Evaluation Result on Usefulness of the Electronic Binder

In the 1st survey, half of the respondents who had ever accessed the electronic binders evaluated the electronic binders positively. The other half, however, did not. But in the 2nd survey, 90 % people evaluated the electronic binders positively. We think this result come from permeation of the quality system and acquisition of the ISO9001 certification.

Of those who evaluated the electronic binders positively, we asked why they found the electronic binders useful. We did this by asking them to select from the following six choices (A) - (F)(multiple choices were permitted). Item (E) was enumerated in the 2nd survey only.

(A) documents and quality records of projects or sub-projects that a respondent belongs to have been ordered by the electronic binders and can be shared/accessed easily.

(B) documents and quality records of other projects or sub-projects have been ordered by the electronic binders and can be shared/accessed easily.

(C) understanding of the ISO9001 standard has been facilitated.

(D) preparing for internal quality audits or ISO9001 formal examinations could be done efficiently.

(E) internal quality audits and ISO9001 formal examinations could be done smoothly.

(F) the other (the reason can be described free)

The result is shown in Fig. HAZEYAMA 4. In both surveys, (A) `documents and quality records of projects or sub-projects that a respondent belongs to have been ordered by the electronic binders and can be shared/accessed easily' was the most answer (around 75 %), and then (B) `documents and quality records of other projects or sub-projects have been ordered by the electronic binders and can be shared/accessed easily' was the second most answer (around 65 %). The percentage is also almost same in both surveys.

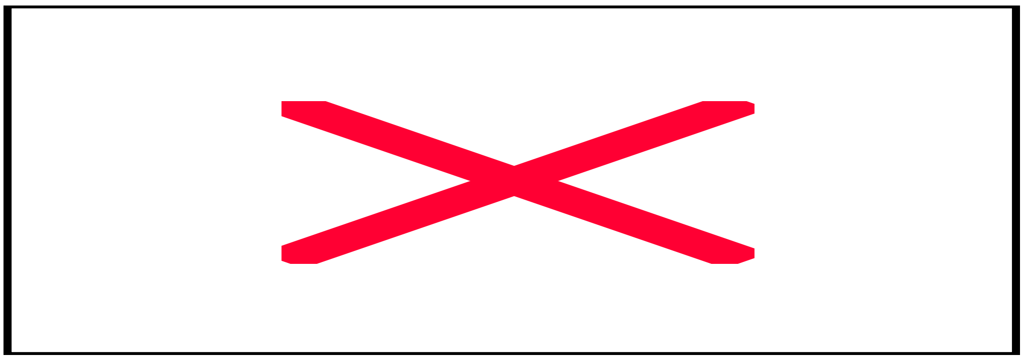


Fig. HAZEYAMA 4: Reasons of Positive Usefulness of the Electronic Binder

We also asked those who had evaluated the electronic binders negatively why they had evaluated them that way by selecting from the following four choices (multiple choices were permitted).
the benefit was not worth the effort.
necessary information has not been found out.
information has not been latest.
the other (the reason can be described free)
The result is shown in Fig. HAZEYAMA 5.

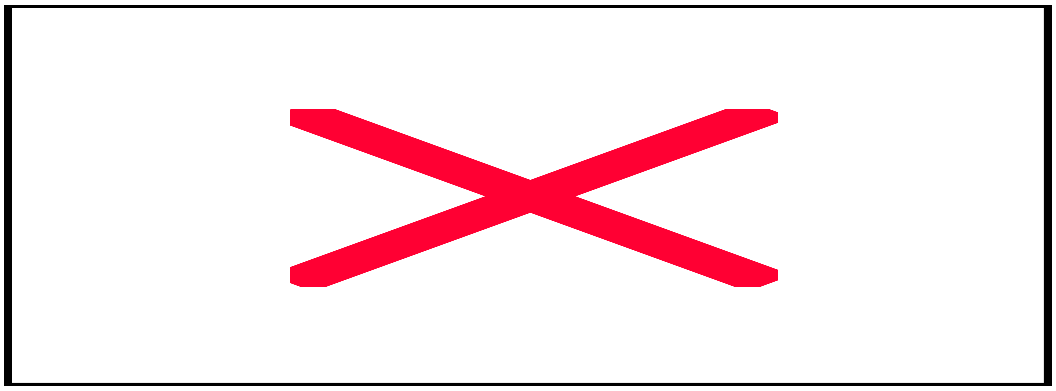


Fig. HAZEYAMA 5: Reasons of Negative Usefulness of the Electronic Binder

In the 1st survey, the most common answer was that the benefit was not worth the effort. The main reason was that almost all documents and quality records registered in the electronic binders had been distributed to the appropriate members by E-mail or that the members had been able to access those documents and quality records by other means (for example, by book-marking necessary documents and quality records/pages of their own on a WWW browser). Therefore it turned out that the members have not necessarily accessed such documents and quality records via the electronic binders (each person manages his/her necessary documents and quality records for him / herself). But as the number of documents and quality records increases, they are used to access documents and quality records via the electronic binder system.

Conclusion

This paper has described a methodology of software process innovation in NEC and a case study of a software development organisation in NEC for promotion of an ISO9001-based quality system using the Web computing and its result. It is important to manage documents and quality records from the point of view of ISO9001 requirements and for software development projects to do so. We think it is necessary for the ISO9001 promotion to define documents and quality records to be managed in the company / organisation. Once documents and quality records are defined, the templates of the electronic binders can be standardised in the company / organisation level. If a company / organisation can establish this kind of framework, people only link to the registered documents or quality records from the electronic binders in the operation stage. The maintenance cost is very small, therefore the return on investment is substantial.

The organisation in this case study has got the ISO9001 certification and therefore

accomplished the first stage of the innovation methodology and now stands in the improvement stage. The organisation is improving the quality system based on the corrective actions and quality goal setting and its follow-up. Some actions are taken as follows:
Improvement of SI (System Integration) and configuration management processes: these processes are very important, in general, especially for large-scale projects.
Collecting a variety of quality data, its analysis (drawing trend curve of faults, root cause analysis of faults, etc.), and feedback to quality goals and management process.
Writing post mortem report after project completion.

For the evaluation of projects from the viewpoint of business (cf. ISO9001 clause 4.1), the organisation now starts to use a framework of the Quality Award.

Such information can be seen from the electronic binder system. The electronic binder system is therefore evolving as an information infrastructure for software projects.

Acknowledgement

The first author would like to thank Mr. H. Oka and Mr. T. Fujimura for support of this work. The authors also would like to thank the anonymous referees for their useful comments to improve this paper.

References

Humphery W. S., Managing the Software Process, Addison Wesley Publishing Co., Inc., 1989.

Honda K., Sunazuka T., and Miyashita Y., Software Process Innovation Methodology - Multiple Approach Including ISO9001, Maturity Model and QC Technique: in the NEC Research & Development, Vol. 38, No. 1, pp.96-104, 1997.

ISO9001: Quality Systems - Model for quality assurance in design, development, production, installation, and servicing, 1994.

Kajihara J., Amamiya G., and Saya T., Learning from Bugs, IEEE Software, Vol. 10. No. 5, pp.46-54, 1993.

Mizuno, Y. (ed.), Software Total Quality Control - NEC’s SWQC -, Nikka Giren, 1990 (In Japanese).

Paulk M. C., Curtis B., Chrissis M. B., and Weber C. V., Capability Maturity Model for Software, Version1.1, CMU/SEI-93-TR-024, February 1993.

The PASS Software Process Improvement Experiment in Hungary (PASS EP 21223)

János Ivanyos
MemoLuX Ltd., Hungary
Dr. Miklós Biró
MTA SZTAKI, Hungary
Dr. Richard Messnarz
ISCN, Ireland

Introduction

The PASS (Pay-roll Accounting and Settlement System) project is the first Central and Eastern European ESSI PIE project directly supported by the European Commission. The PASS project is carried out under the ESSI initiative with the financial support of the Commission of the European Communities under the ESPRIT Programme EP21223.
The PASS (Pay-roll Accounting and Settlement System) project started as a new business project of MemoLuX. Its business purpose is to develop a modular, platform independent, integrated networked software system satisfying the functional requirements of EU standards in public accountancy and applicable for the Hungarian as well as for the international market. The system provides direct service among Employers, Employees and Banks. The PASS project is the baseline project for the Process Improvement Experiment.
In the PIE the quality of MemoLuX´s development process was enhanced to become well defined and predictable, and during the dissemination supported by ISCN, this PIE is used as a master example to adapt Eastern European processes to the high quality norms of Western Europe, this way facilitating the integration of Eastern Europe into a joint EU in the long term. Objectives and expected results are improving the control of the development process (QA Unit, structured system analysis, improved testing process, efficient project planning, ISO 9001 documentation), raising the maturity level (CMM) to 3 and achieving compliance with ISO 9001 requirements at this high level of maturity.

Background Information

On March 16, 1995 The IT Program (Esprit) published its sixth call for proposals under the Fourth Framework Program. The changing policy of the European Commission brought the opportunity for CEEC participants to apply directly to the Commission with proposals and under the Process Improvement Experiments (PIE) Tasks MemoLuX produced an accepted proposal. The uniqueness of the situation came from the fact that it was the first time that a proposal was submitted with a non-EU Prime User and Co-ordinator. Nevertheless, the call allowed the formation of a consortium from one country only, which caused legal difficulties during the contracting period. The legal issue was solved by the involvement of ISCN as associated partner of the project and finally the contract came into force by the end of June 1997.
See **Figure 1. The new features of process improvement under the ESSI PIE project** in the Annex.

Objectives

Since MemoLuX Ltd. is steadily growing and is managing larger and larger and a greater number of projects, MemoLuX has to achieve better control of the software development process, a quantitative view of the production process, and higher credibility among Hungarian and EU customers through an improved BOOTSTRAP maturity profile and an increased level of compliance with ISO 9001. The aimed maturity level 3 is fairly high on the international scale. The progress is monitored according to a well planned quality measurement process and corresponding tasks are scheduled as separate workpackages.
See **Figure 2. Specific measurable objectives** in the Annex.

Involved companies and their roles

MemoLuX Ltd.

MemoLuX Ltd., (URL: http://www.memolux.hu) a Hungarian private company with professional experience is a service provider in finance and public accountancy, management organization, software development and information system engineering.

ISCN The International Software Consulting Network

ISCN (URL: http://www.iscn.ie) offers professional services in the fields of process analysis, process modeling, process and product measurement, and practical experience with the installation and performance of improvement projects.

MTA SZTAKI (Computer and Automation Institute of the Hungarian Academy of Sciences)

MTA SZTAKI (URL: http://www.sztaki.hu) is the largest research institute of the Hungarian Academy of Sciences. Contract-based target research, development, training and expert support for domestic and foreign industrial, governmental and other partners have been key activities of MTA SZTAKI since its foundation (1972).
The participating organizations' roles:

MemoLuX Ltd	co-ordinator/prime user, baseline project , improving the software development process, dissemination
ISCN	associated partner, requirements for control & measurement of results, preparing all results as best practice reports, project monitoring, dissemination of results for EU
MTA SZTAKI	subcontractor, consultation, training, implementation of the measurement plan, BOOTSTRAP methodology

Starting scenario

Before starting the ESSI PIE project, significant improvement of the software development process had been achieved. During the period from November 1995 to March 1997 MemoLuX performed improvement actions, the software development process improved significantly by the work of MemoLuX's own staff, and the results were approved by the second BOOTSTRAP assessment. The maturity level of MemoLuX rose above the repeatable process level (2.5 CMM score achieved). The succes of the former project led MemoLuX to a better starting position than the one it had when the EU Software Best Practice started.

The BOOTSTRAP assessment was carried out by MTA SZTAKI, the only Hungarian licensed assessor, in order to clarify the strengths and weaknesses in the software development process at MemoLuX. This gave the following conclusion:
The maturity level was 2.5 in the SPU (Software Producing Unit) global environment and the same score was measured in the selected SPU project. The main strengths and weaknesses as deviations from the average are outlined based on BOOTSTRAP attributes in **Figure 3. Strength and weaknesses at starting phase** in the Annex.
Looking at the status of software engineering practice at MemoLuX, the quality management of IT projects needed enhancement. Improvement steps, to be carried out first with the guidance of Subcontractors, were planned in several areas:
Organization

- setting up a Quality Management Unit
- setting up a Project Management Unit
- providing quality management training for the project management and staff

Methodology

- preparing the Quality Manual for the software development activities
- establishing a standardized and documented software development process
- introducing standards to software product documentation
- introducing standardized testing methods (including metrics, documentation)
- creating and using metrics for measuring project progress, collecting and analyzing progress and product metrics

Technology

- in the software development unit:
  - ❍ unified methods and tools for configuration and change management,
  - ❍ planning tools possible with graphical representation for the planning model, software development plan and schedules.

The business purpose of project PASS is to develop a modular, platform independent, integrated networked software system satisfying functional requirements of EU standards in public accountancy at a European business level and applicable for either small or medium or even large enterprises working for the Hungarian as well as for the international market. The system provides the direct service among Employers, Employees and Banks for applying bank transfers. More and more companies are willing to outsource their pay-roll accounting. However, the PC based pay-roll account systems available in the Hungarian market are not suitable for Employers having several thousands of Employees. The need for pay-roll systems with 100.000 items is emerging. The creation of a nation-wide networked software system for supporting up-to-date banking services in Hungary is a kind of solution for solving these problems. The result of PASS project is a nation-wide IT system for pay-roll accounting and settlement, providing services at a European level in three sectors of economic life: small, medium or large enterprises, banks and active population.
To complete the phases of the PIE and the baseline project new organizations were defined. These were the project organization for PIE, the project organization for the baseline project, Steering Committee, Project Board, QMU (Quality Management Unit) setup for MemoLuX, establishment of the risk management team, etc. The next step was to define the roles, the persons and the organizations involved in each workphase.
Since the problem of who does what was solved by defining the roles and the associated responsible persons, we had to make decisions on how we were going to put our thoughts together. At the premises of MemoLuX an FTP server was set up and all the released and work documentation are stored there. The server is always on-line. The system administrator distributes the e-mails from here to everybody on the list. Additionally, the useable software standards were fixed for scheduling and documentation, common templates were made, reviewed, accepted and distributed.

Expected outcome

MemoLuX Ltd. is steadily growing and is managing more and more larger projects. While introducing best practices, the company can effectively manage a large number of projects in the future by reusing the lessons learned.
In our opinion, any company in a situation similar to that of MemoLuX could benefit from the PASS process improvement results which show how to start improvement programs in those sectors that are of critical importance in businesses between Eastern and Western Europe.
MemoLuX is able to establish a stable and predictable development process, Subcontractors support MemoLuX in effectively implementing and using the new quality system, and ISCN ensures a European wide dissemination of results plus consulting about how to measure and control process improvement. ISCN ensures that the results will be discussed in a broader community in the EU via WWW and conferences to make the Hungarian efforts and results visible as well as to enable a feedback loop between EU PIEs and this Hungarian PIE.
The anticipated benefits of the better control of the software process and of achieving a quantitative view of the quality of the system/product are the ability to make decisions based on quantitative data resulting in higher credibility for Hungarian, EU, and other customers through the improved BOOTSTRAP maturity profile and increased level of compliance with ISO 9001.

Implementation of Process Improvement

MemoLuX decided to improve its development process. This Process Improvement Experiment project gave MemoLuX a good opportunity to evaluate new methods, procedures and tools in a real life environment to model their processes and to implement a quality system.
Nowadays the formal modeling of processes is gaining increasing interest in the field of analyzing organizations with respect to the quality of their products, productivity and efficiency. The process models are the basis for improvement actions and comparisons.
The development of a process model is a process itself. This process had to be adapted to the specific needs of MemoLuX. The level of detail to be specified depends on the level of the development process (see BOOTSTRAP assessments), the available computer system configuration, skills and experience of the personnel, and the size of the organization.
Four scenarios were selected for trying out how to build a workflow and for using these experiences:

Project Planning Scenario

Review Modeling Scenario

Configuration Management Scenario

Testing Scenario

After the technical implementation of the selected scenarios into the LBMS PE tool, there were two main streams of work performed. At first the implemented workflows were converted to working instructions of the ISO 9001 Quality System Documentation, and after the coaching and training steps, the usage and data collection started within the baseline software development project as well.

The usage of workflows and quality system documents started during the baseline project initiation (December, 1997). Due to the decision after the investigation of the results of the mid-term self assessment, the overall implementation of the ISO 9001 Quality System for the whole IT organization started in April, 1998.

In the PASS project the quality of MemoLuX's development processes were largely enhanced to become well defined and predictable, so the measurement plan supported the collection and presentation of a set of objective data for illustrating success and / or failure, and the lessons learned.
Performance metrics were applied to show the progress in the field of business performance and software maturity. Workpackage metrics were applied to give quantitative feedback. The measurement plan implementation described how to apply the metrics given in the measurement plan to the workpackages of the PIE and to the baseline project. The measurement plan implementation included measurement issues related to all workphases of the PIE and the baseline project.
Business performance metrics were taken before and after the completed PASS project and are compared in **Figure 4. Business performance metrics** in the Annex.
Maturity metrics were taken three times: at first in 1996, at mid-term of the project by performing BootCheck self assessment and finally Bootstrap assessment was done in October, 1998. The overall maturity level and the

maturity level of each attribute are compared to the project’s final Bootstrap assessment in the standard Bootstrap form. See **Figure 5. Bootstrap maturity profiles** in the Annex.

Organization

To complete the phases of the PIE and the baseline project new organizations were defined. These were the project organization for PIE, the project organization for the baseline project, Steering Committee, Project Board, QMU (Quality Management Unit) setup for MemoLuX, establishment of the risk management team, etc. The next step was to define the roles, the persons and the organizations involved for each workphase. See **Figure 6. Organizational structure of the PIE** in the Annex.

Technical environment

MemoLuX made an evaluation process to select the most suitable process modeling tool. The LBMS Process Engineer tool is a system for developing systems. It consists of a set of processes for planning, managing, and developing Information Systems and gives a technology for automating the use of the system. LBMS offers a product that not only provides an extensive library of best practices, but can help an IT organization to capture its own existing intelligence as the organization’s best practices. The components of LBMS Process Engineer ordered by MemoLuX are the Process Manager with the Process Library and the Project Manager. The Process Library stores best practices of which MemoLuX’s repeatable processes can be built, making a standard for development and raising the expertise of the entire organization. Process Management is the method to capture, deploy, execute and improve best practices for continuous improvement. Applying Process Management new processes can be authored, or the best practices from the Process Library can be customized to satisfy MemoLuX’s needs so they can become organization standards. The Project Manager provides the ability to generate detailed project plans based on the processes, define and store information on the progress of deliverables, roles and resources and apply metrics and estimating models.
See **Figure 7. LBMS Process Management**

Training

Training and participation on conferences are basic activities of the project. Specific training was performed regarding the usage of the LBMS Process Engineer tools, implementation of the four selected scenarios and the ISO 9001 Quality System. External assistance was provided by MTA SZTAKI and the ISO 9000 consultant Qualyfore. During the training period the permanent improvement of quality system deliverables and the internal dissemination were given priority due to the concept of ISO 9001 preparations.

Role of the consultants

ISCN as the associated partner, provided requirements for control & measurement of results, prepared all results as best practice reports, contributed to project monitoring and dissemination of results for EU. MTA SZTAKI as subcontractor, provided consultation, training, implementation of the measurement plan and made BOOTSTRAP assessments. Qualyfore as ISO 9000 consultant subcontractor provided ISO 9001 training on the improved quality system deliverables.

Phases of the experiment

As part of the project initiation, 6 main stages (see **Figure 8. Phases of the experiment** in the Annex) were set up covering the workpackage structure to provide checkpoints for project progress. Due to the fact that there are strong connections between the baseline software development tasks and PIE tasks in measurement and quality monitoring issues, the set up and implementation of the Quality System were performed before completing the system planning phase of the PASS development, so only the project initiation of the baseline was completed at the first stage.
At the endstage review of the first stage performed in December, 1997 corrective actions were indicated in time, during the detailed planning and scheduling of the next stages:

- resources were re-allocated from the project management task to the use-steps of the scenario development to support actual try-out in the baseline project
- MemoLuX re-allocated management days to technical days because the try out needs much more technical resources
- a software circle as a feedback mechanism was founded
- all intermediate results of each scenario and quality development were discussed in this circle.

Corrective actions succeeded. Two critical problems were solved. On the one hand the full ISO 9001 Quality System documentation was completed in compliance with the redefined process workflows, on the other hand internal dissemination connected with continuos improvement of the deliverables was working due to the new feedback mechanism.
After the completion of the ISO 9001 Quality System documentation based on the implemented quality scenarios and successful training and coaching period at the closure of the second stage, the decision was made to complete the implementation tasks of the Quality System in the third stage. This meant the overall introduction of the Quality System in the whole IT organization which had been originally planned at later stage. This meant extra effort for the IT staff during the third stage so the baseline project was suspended for one and a half period. The success of the performance of this stage was measured by the mid-term self assessment, and after internal audits, a successful ISO 9001 certification was achieved.
On the basis of the results of the first three stages, the baseline project schedule was updated. The parallel activities of the baseline and the PIE projects regarding measurement and quality monitoring were performed in stage 4 and 5. The fifth stage ended by the final Bootstrap assessment of the project. The expected maturity level of the IT department was 3.
In stage 6 the external dissemination activities were completed.

Internal dissemination

Based on the results of the mid-term BootCheck self-assessment, the commitment of management helped in disseminating the results within the company.
Due to the repeatable actions carried out and the transferable results of the process improvement, the introduction of the ISO 9001 Quality System came into force in April instead of the planned date of November, 1998. This means that the results of the PIE were not only disseminated at mid-term, but influenced the whole IT organization of MemoLuX by the overall introduction of the Quality System.

Results and Analysis

Technical

The technical implementation of the scenarios and the customized software development process contributes to the replicability of the PIE results due to the availability of the Process Library and the direct and comfortable usage of the workflows in project planning and scheduling.
The implemented quality scenarios were directly used for creating the working instructions of the ISO 9001 Quality System documentation. This was the key issue to achieve compliance with the ISO 9001 requirements on the basis of a high capability maturity level. The direct result was the extremely short period of the overall introduction of the Quality System, which is one of the main experiments of this PIE.
The complaince of the software process related quality scenarios with the chapters of the ISO 9001 Quality System documentation was defined as follows:
Project Planning Scenario was implemented as the working instruction of Design Control in order to establish and maintain documented procedures to control and verify the design of the - software - product.
The working instruction implemented from the Configuration Management Scenario was referred at the chapters of Document and Data Control, Product Identification and Traceability, Inspection and Test Status, Control of Nonconforming Product, Handling, Storage, Packeging, Preservation and Delivery.
Review Modeling Scenario and Testing Scenario were implemented as the working instructions of Inspection and Testing. See **Figure 10. Inspection and testing during software development.**
In most of the workpackages starting the work meant making a plan of performing the tasks. Before the beginning of the completion of this plan, it had to be preceded by a review of the plan to make sure that the goals and objectives were correct and that the requirements would be met. This is called the **design review**. At appropriate stages of the software development project, typically at endstage, formal documented reviews of the project plan were planned and conducted.
In the PASS PIE project the products of each workpackage were documents, so the **document review** path was followed using the Review Modeling Scenario.
In the case of the PASS baseline project the products of the workpackages were either software or documents, so the appropriate path (of **Figure 10. Inspection and testing during software development**) was applied to decide the type of test and review to be executed.
The ability of a software was examined by running tests and the test reports were discussed in the appropriate reviews, these are called **test case reviews**.

Depending on the life cycle phase of the software product the following tests and reviews were executed:

The **design review** is a review of the software development process.

The **test strategy review** is a review of the tailored test startegy.

The **module test** is completed on the program modules, followed by the **module test case review.**

The **integration test** is completed when the complete modul is ready and proves that the parts of the product co-operate correctly, followed by the **integration test case review.**

The **system test** is completed on subsystems or systems, followed by the **system test case review.**

The acceptance test is usually done in the presence of the customer followed by the **approval review,** where both software and documentation are checked.

The main quantitative result of the project regarding scheduling is the two-month delay in baseline development compared to the one-month introduction period of the ISO 9001 Quality System. The software industry based expectation for the introduction was 6 months. This means that the formerly achieved process improvement maturity level 3 causes delay in the first project, but this delay is comparable to the introduction period of the ISO 9001 Quality System. The clear advantage is the extremely fast achievement of compliance with ISO 9001 requirements for the whole IT organization.

Business

The following goals, as leveraging effects on business performance, were measured at the end of the project:

Extension of business activities to financial services (number of financial transactions)

More competitive service charges due to decreased production costs (price per employee)

Increased volume of pay-roll services (number of employees processed per month)

The PIE has a direct leveraging effect on business performance, achieved by improving the control of the development process measured by the increased maturity level of the organization’s software development capabilities.

The maturity results of the experiment were measured even at midterm (April, 1998) by BootCheck Self Assessment Tool. BootCheck is a joint initiative sponsored by ESI (European Software Institute) and Bootstrap Institute. The aim of BootCheck is to enable individual Software Producing Units in an organization to make quantified assessments of their software capabilities, and to use this information as a key element in continued process improvement programmes.

According to the Bootstrap overall maturity profile, SPU Maturity level was 2.75, near the Defined level, which means that the software process is documented, standardized, and integrated into a standard software process for the organization. Organizational profile was also rated at 2.75, with the relative weakness of resource management. From the methodology point of view the process related functions scored at 3.75, life cycle functions at 2.5 and life cycle independent functions at 2.75.

The final maturity metrics are produced by the Bootstrap assessment performed in October, 1998. The main improvements compared to the mid-term status are achieved in resource management, life cycle and life cycle independent functions due to the running Quality System and further improvements in development cycle modeling and architectural design. See **Figure 5. Bootstrap maturity profiles** in the Annex.

Due to the shorter - one instead of a six-month - introduction period of the ISO 9001 Quality System, the certification was achieved half a year earlier than it had been expected. The publication of the ISO 9001 certification towards MemoLuX’s clients met with their appreciation, and helps MemoLuX utilize the advantage of having higher credibility in the eyes of Hungarian, EU and other customers.

Organization

The Quality Assurance Unit is the really new structure element appearing from the improvement of the quality processes. It is important, that the Quality Management organization is separate from the software development units. Project schedules are always tight, and project managers should not be worried about inadequate test plans, human factor problems, or documentation errors. If the size or complexity of a project requires, quality management functions can be dedicated to a person at project level as well.

The Quality Manager or the member of the Quality Assurance Unit works closely with the developement organization. It is relevant to understand every part of the process customized for the development activities.

Culture

The implementation of the whole ISO 9001 Quality System regarding the IT department of MemoLuX started in April, 1998 after continuous training and consultation on the materials. The quality policy was fully understood and accepted by the whole staff. The fulfillment of ISO 9001 requirements for the software development organization aiming at a defined maturity level at the same time was a great effort, but a real challenge for the well educated staff of MemoLuX.
The earlier resistance to adopting the new working instructions was overcome by the continuous involvement of the staff members through the so called Software Circle discussion forum and after the introduction of the Quality System two immediate internal audits helped the staff to get the correct adaptations. The compliance was measured by the ISO 9001 certification within an extremely short time.

Skills

Regarding the Quality Management personnel it is very important to have sufficiently experienced and knowledgeable people. In our practice the Quality Manager, as a system design expert with quality skills, can monitor whether the methods and standards used by the software development experts are consistently and correctly applied. The following skills are necessary for doing this job properly:

knowledge of quality control procedures

knowledge of the software development process

knowledge of statistical methods

and it is very important to have the ability to deal effectively with people in controversial situations.

Key Lessons

This section summarizes the key lessons we have learnt from undertaking the experiment.

Technological point of view

One of the fundamental principles of the BOOTSTRAP software process improvement methodology is that before any investments are made in technology, the methodological questions on how to build solutions have to be answered, the methodological solutions to be institutionalized have to gain organizational acceptance and be compatible with the existing or improved processes of the organization.
This principle conforms to the philosophy promulgated by Watts Humphrey in his inseminating work on Managing the Software Process where he writes the following:
"They are thus prone to embrace some magic technological "silver bullet" that will painlessly solve all their problems. While technology is important and is the long-term key to improving the software process, it is rarely of much help in solving problems that are not precisely understood. Since most people object to having someone else's solutions to undefined problems imposed on them, the unplanned introduction of technology generally causes resistance to change."
The above BOOTSTRAP priorities are usually summarized by the following formula:

$$O > M > T$$

where O stands for Organization, M stands for Methodology, and T stands for Technology.
Armed with all the above wisdom, we still fell in a double trap. Since there was a budget allocated in the project for buying a tool, we wiped out our wisdom and carefully selected the LBMS tool before institutionalizing the necessary processes. Then, with the tool at hand, we could not wait plunging into learning it, expecting that it would provide us ready-made solutions for all of our problems. Nevertheless, we had to realize the facts of life quickly. Not all of the necessary processes were readily usable in our case in LBMS.
At this point, remembering the wisdom, we returned to defining both the planning and review processes independently from the tool. On the one hand, this approach proved well to be necessary for the review scenario. On the other hand, we also invested a lot of work into the planning scenario until we discovered that in this case, the model coming with LBMS was actually fully applicable. This is the point where we came to appreciate that LBMS was not simply a tool but also an experience library to which the above wisdom does not directly apply.
The lessons from the above experience are the following:
1. Do not yield to the temptation of immersing into the use of however expensive tools before having analyzed the real problems and processes the tool is intended to solve. This was the reformulation of the above wisdom.

2. Today, most products on the market, that are positioned or considered as tools, go much further than that. They include a library of templates or experiences which may well be more valuable and have a validity independent from the tool itself. The above wisdom does not apply to these libraries which are most of the time worth being examined before starting to analyze our own problems.
3. After examining the libraries, do not yield to the second temptation of immersing into the use of the tool. Listening to the wisdom, examine your real problems and processes and try to critically match the solutions offered by the library to them. If the match is not perfect, try to adapt first and ultimately develop your own solutions. This is the point where the use of the tool may become valid and also appreciated.
The above lessons allow us to extend the scheduling priorities the BOOTSTRAP methodology is based on. Using the formula, the priorities look like the following:

$$L > O > M > T$$

where L stands for Library of experiences or templates. The importance of the extension lies in the fact that the cornerstone of the model is still O while the so called tools include L, M, and T.

Business point of view

It was the development of MemoLuX's organization in compliance with ISO 9001 requirements and not the certification itself which was an objective of the PIE. We recognized in time however that the improvement of our maturity level brought us within short reach of the ISO 9001 certification level. We decided at this point to reallocate resources so that we can achieve this business objective. The consequences of our decision were the following:
- The baseline project was delayed by two months which caused a minor rescheduling of the PIE itself.
- The maturity level previously improved in the framework of the PIE allowed us to reach ISO 9001 compliance and actual certification within one month as compared to the originally expected duration of 6 months.
- The higher credibility due to our ISO 9001 certificate brought immediate business benefits significantly earlier than originally anticipated in the PIE.
- Process improvement could be perfectly continued after the certification.

The lessons from the above experience are the following:
1. The approach of considering the improvement of the maturity level as the principal objective and the achievement of ISO 9001 certification as a side-effect is valid from the efficiency point of view.
2. Even if ISO 9001 certification is not the principal objective of process improvement, it may be worth capitalizing on its high recognition by allocating appropriate resources to its achievement at the right stage during the process improvement project. The business benefits may well outweigh the effect of the resulting delays in the process improvement itself.
3. The ISO 9001 certification does not only have direct business benefits. According to international experiences, there is usually a significant decline of attention towards the quality system after the certificate is granted. The approach of considering certification as a side-effect of overall process improvement not only helps avoiding this trap but the quick success even has a spurring effect on the whole IT organization regarding further process improvement.

Strengths and weaknesses of the experiment

The main strength of the experiment is that most of the weaknesses could be corrected. Due to the strong project monitoring, the need for corrective actions was recognized in time (at the closure of the first stage).
- resources were re-allocated from the project management task to the use-steps of the scenario development to further actual try-out in the baseline project
- a software circle as feedback mechanism was founded
- all intermediate results of each scenario and quality development were discussed in this circle.

The corrective actions resulted in success. Two critical problems were solved. On the one hand the full ISO 9001 quality system documentation was completed in compliance with the redefined process workflows, on the other hand internal dissemination connected with continuos improvement of the deliverables worked due to the new feedback mechanism.
Further lessons derived from recognized and corrected weaknesses are summarized below.
The quality system and the scenario development were running in parallel, however at the beginning no links were taken into account. A quality system consists of the Quality Manual, Procedures, and detailed Working Instructions for the procedures. The scenarios should have been aligned with the Procedures and Working Instructions in the Quality Manual, or at least there should have been a mapping.
The introduction of a scenario (e.g. review scenario) was done following a predefined series of steps

- analyzing the model
- implementing
- coaching and training
- use
- feedback (data collection)

There was however a misunderstanding of the "use" step. In order to overcoming such misunderstandings, a software circle had to be created, consisting of the experts and the staff that in future would have to work with the results produced by experts. The software circle had a meeting every two weeks for 2 hours. The result of each step was presented to this circle, opinions were discussed, and feedback was recorded. Without this complementary action, it is questionable whether the results (new workflows) would have ever been accepted by the staff.

Conclusions and Future Actions

MemoLuX, with its partner and subcontractor, has achieved its main goals. It significantly improved the control of the development process, increased the maturity level, completed the ISO 9001 quality system documentation and made headway in running the quality system. The dissemination of the experiment contributes to the replicability of these results for other SMEs from the region which is not only a potential marketplace, but also a strong battlefield for competition.
Future actions planned and designed in new projects being set up upon compilation of the PIE. These are the Process Improvement Project and the Quality Improvement Project based on the achieved results detailed in this article. Running the quality system helps us to control the development of our software and quality processes. New scenarios are planned to be defined and implemented into the process library as it has been successfully done with the four basic quality scenarios.
MemoLuX is going to utilize the advantage of having practice in quality system implementation as working together with Hungarian, EU and other customers to support them to achieve improved software maturity and increased level of compliance with ISO 9001 especially in fastly growing software development organizations.

Glossary

BootCheck	BootCheck Self Assessment Tool sponsored by ESI and Bootstrap Institute
BOOTSTRAP	European Software Process Assessment and Improvement methodology developed by an ESPRIT project.
CASE	Computer Aided Software Engineering
CEEC	Central and Eastern European Countries
CMM	Capability Maturity Model
ESI	European Software Institute
ESSI	European Systems & Software Initiative
EU	European Union
ISCN	International Software Consulting Network
ISO	International Standards Organization
IT	Information Technology
LBMS	LBMS Process Engineer Tools developed by LBMS Inc.
OMFB	Hungarian National Committee for Technological Development
OOP	Object-Oriented Programming
PASS	Pay-roll Accounting and Settlement System
PIE	Process Improvement Experiment
QA	Quality Assurance
QMU	Quality Management Unit
SPU	Software Producing Unit
SQA	Software Quality Assurance
WWW	World-Wide Web

References

[1] Biró M.; Feuer É.; Haase V.; Koch G.R.; Kugler H.J.; Messnarz R.; Remzsõ 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.

[2] Walk K., Messnarz R., Object Oriented Modelling of Work Processes. In: Proceedings of the ISCN'96 Conference on Practical Improvement of Software Processes and Products (ed. by R.Messnarz). (International Software Collaborative Network, Brighton, 1996) pp. 264-287.

[3] Biró,M.; Feuer,É; Ivanyos,J. Process Improvement Experiment 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.

[4] Biró,M; Remzsõ,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)

Annex

Figure 1. The new features of process improvement under the ESSI PIE project:

MemoLuX role as the prime user of the software development activities (Payroll Accounting and Settlement System chosen as baseline development project for process improvement)

Starting the practice from higher level of maturity model (CMM score was about 2.5)

Consulting work on quality issues is given by MTA SZTAKI as subcontractor of the project

Utilizing EU funds for investing in technology (LBMS Process Engineer Tools)

Implementation of measurement procedures in the project

Connection to EU dissemination activities by ISCN from the very beginning of the project

Figure 2. Strength and weaknesses at starting phase

	Strength	**Weaknesses**
Organization	management responsibility	quality system
Methodology	process description process measurement operation and maintenance	testing integration architectural design risk management
Technology	communication user requirements	CASE tools

Figure 3. Structure of measurement

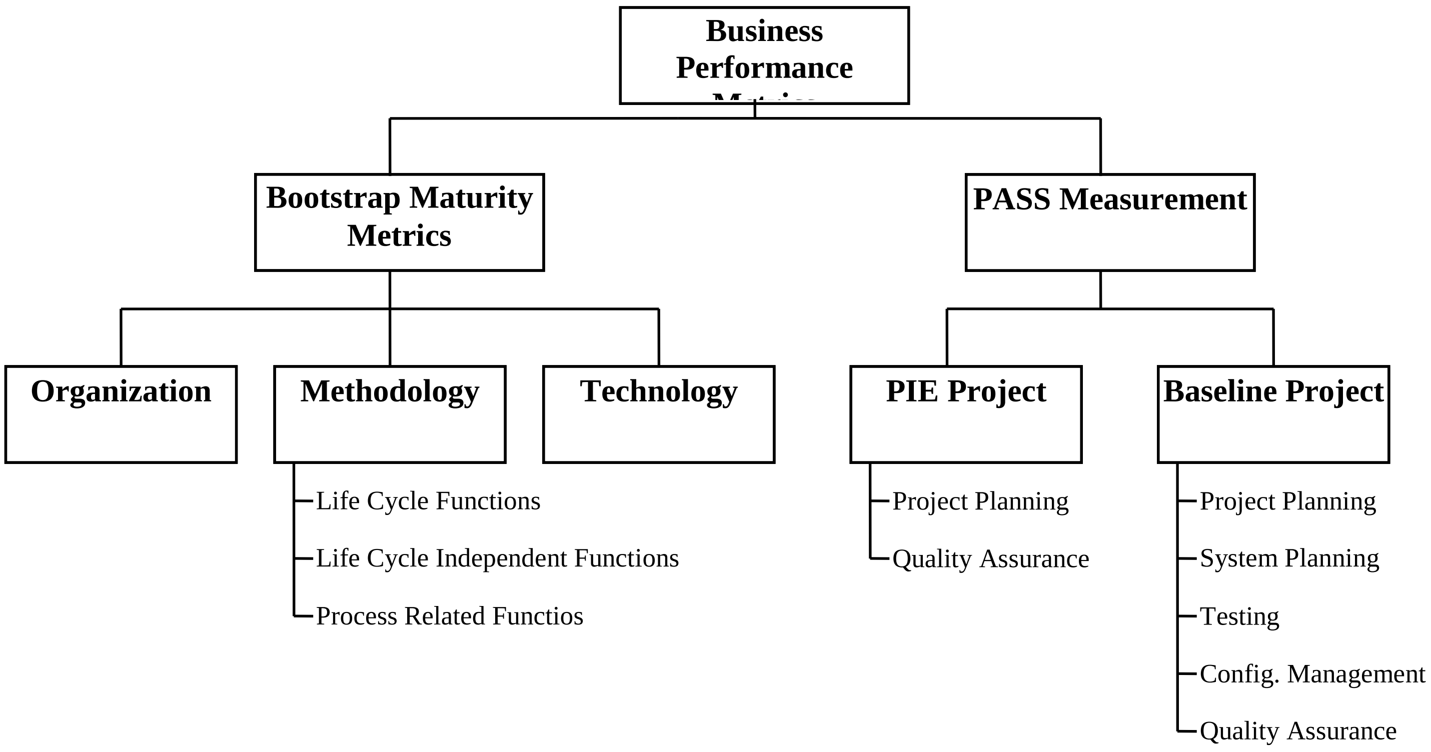


Figure 4. Organizational structure of IT department

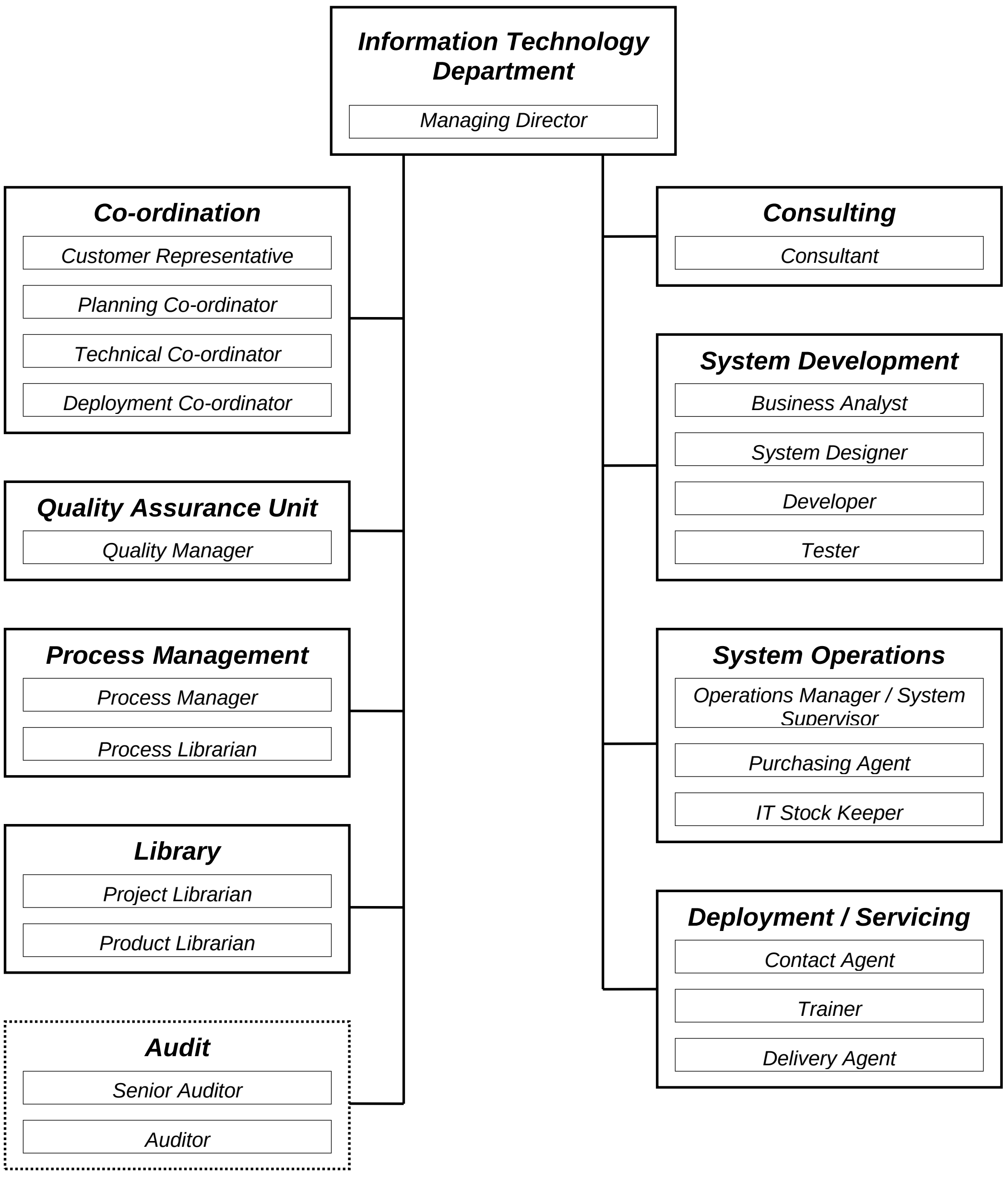


Figure 5. Criteria for process modeling tool selection

Product:	LBMS Process Engineer
Supplier:	LBMS Inc.
Local vendor:	KFKI IBIS Kft
Date:	22. July 1997.
Assessor:	MemoLuX Kft.

	Value(0=no,1=partially,2=fully)	Notes	Proportion (1-10)	Result (Value*Proportion)
System Requirements				
Functionality				
Support to improve the organization's development process to reach leveraging effect on business performance	2	definition, feedback, reusablity	10	20
Supporting of ISO 9000 standard system	5		8	40
Built-in templates	2	kernels		
Possibility to link to different databases	1	command line		
Possibility to create ISO compliant documents	2			
Improving the software development process	4		10	40
Best practice templates for all phases of software development	2	process templates		
Functions to support the maintainance of documents (conf. man.)	2			
Project management promotion	4		10	40
Workflow management	2			
Possibility to link to external project scheduler	2			
Logical access level management	0		6	0
Support audit trail	1	version management	8	8
General document management	2		6	12
Document version management (internal, external)	1	internal only		
Possibility to build external documents into the system	1			
Connectivity Considerations				
Possibility to link to external tools	1	OLE	6	6
Possibility to link to Windows 95 applications (Office,WINPROJ,VISIO)	2		8	16
Hardware Requirements	2		8	16
Platform independence	0	Windows 95		
Possibility to integrate the system with our present hw/sw platforms	2	insignificant capacity extension require		
Financial stability of software supplier	4		8	32
Financial stability of international supplier	2			
Financial stability of local vendor	2			
Availability of Complete and Reliable Documentation	1	electronical	10	10
Vendor Support	9		10	90
Installation	2	local vendor		
On-site training	2	local vendor, international supplier		
Error correction	2	international supplier		
Product upgrades	2	int.supplier,local vendor		
Hot-line	1	local vendor		
Source Code Availibility	1	international supplier has the ownership	5	5
Experience in Offering the Product	1	is not used widely in Hungary	8	8
List of Planned Enhancements to the Product	2	continuing broadcast on home page of supplier	8	16
User Reference List	1	international	6	6
Availibility of Demonstration Version	0		8	0
Occasional Price	2		8	16
Final Result				381

Figure 6. Process Management supported by LBMS PE tool

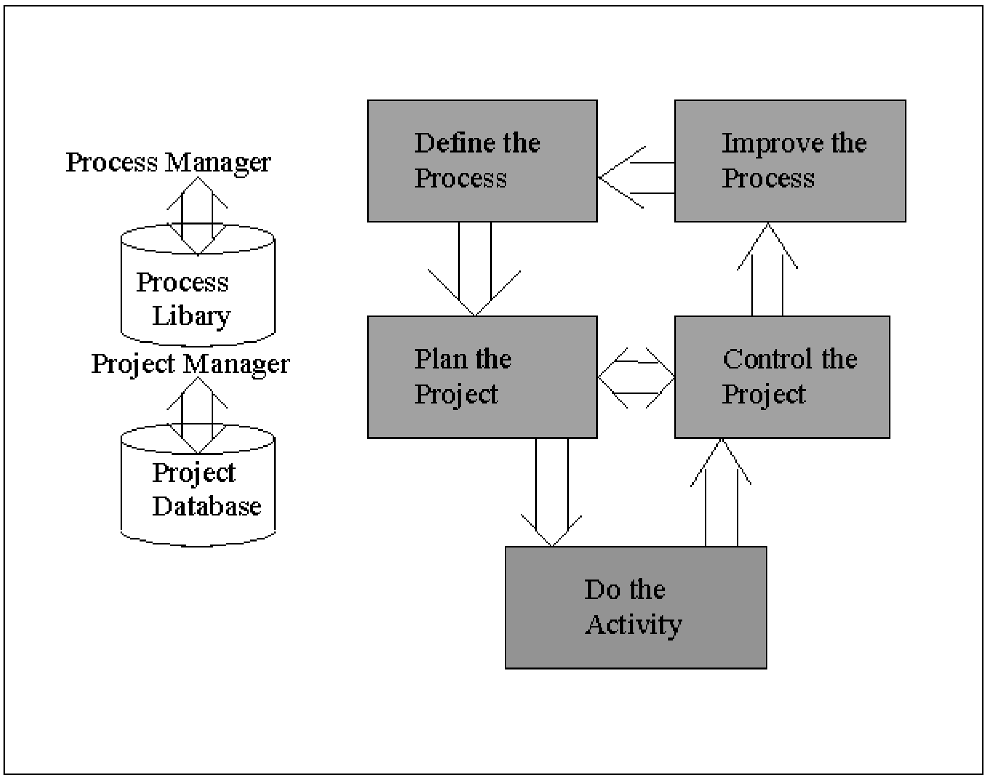
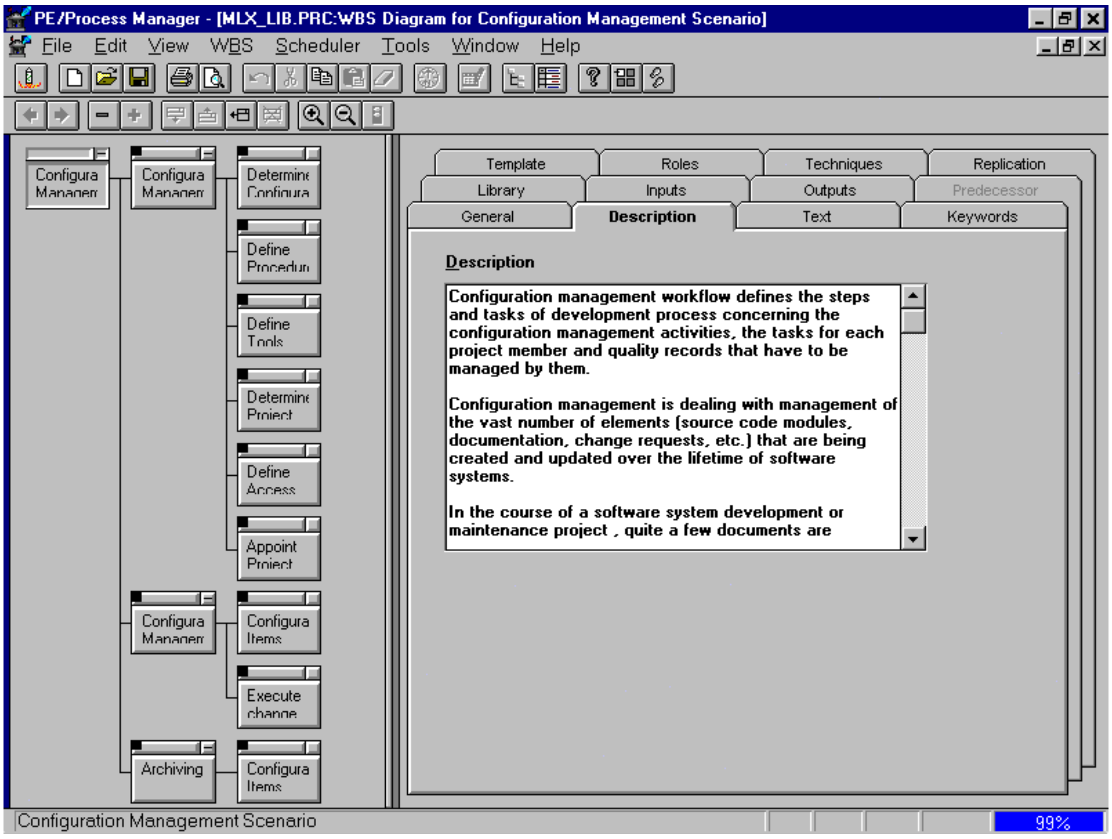


Figure 7. Project schedule

			Milestones			Effort		
Stages	**Project**	**Main Deliverables**	**Planned**	**Actual**	**Deviation (months)**	**Planned (mandays)**	**Actual (mandays)**	**Deviation (mandays)**
1.	PASS PIE	Project Set-up, Tool Purchase Measurement Plan Implementation Quality System Implementation (Project Management, Reviewing)	October 1997	December 1997.	2	313	354	41
	PASS Baseline	Project Initiation Documentation	October 1997	December 1997.	2	40	40	0
2.	PASS PIE	Quality System Implementation (Testing, Configuration Management) Quality System Documentation	January 1998	March 1998.	2	150	253	103
	PASS Baseline	*/System Planning completed in Stage 3/*						
3.	PASS PIE	Midterm Evaluation	April 1998	April 1998	0	59	45	-14
	PASS Baseline	System Plan	February 1998	May 1998.	3	120	315,5	195,5
4.	PASS PIE	*/Baseline Project Activities/*	July 1998.	Oct. 1998.		258	225	-33
	PASS Baseline	Implementation, Testing System Documentation (Basic)	July 1998.	October 1998.	3	410	357	-53
5.	PASS PIE	Quality Assurance Documentation Measurement Documentation BOOTSTRAP Assessment	November 1998.	October 1998.	-1	47	80	33
6.	PASS PIE	Best Practice Report, Dissemination Project Closure	January 1999.	December 1998.	-1	60	51	-9
Summerized values								
	PASS PIE				-1	887	1008	121
	PASS Baseline				3	570	712,5	142,5
Summerized delay / overrun								
	PASS PIE				-5%			13,6%
	PASS Baseline				27%			25%

Figure 8. Implementation of Configuration Management Scenario into LBMS PE tool


PE/Process Manager - [MLX_LIB.PRC:WBS Diagram for Configuration Management Scenario]
File Edit View WBS Scheduler Tools Window Help
Configura Managem
Configura Managem
Determine Configura
Define Procedur
Define Tools
Determine Project
Define Access
Appoint Project
Configura Managem
Configura Items
Execute change
Archiving
Configura Items
Template Roles Techniques Replication
Library Inputs Outputs Predecessor
General Description Text Keywords
Description
Configuration Management Scenario
99%

Configuration management workflow defines the steps and tasks of development process concerning the configuration management activities, the tasks for each project member and quality records that have to be managed by them.

Configuration management is dealing with management of the vast number of elements (source code modules, documentation, change requests, etc.) that are being created and updated over the lifetime of software systems.

In the course of a software system development or maintenance project , quite a few documents are

Figure 9. Inspection and testing during software development

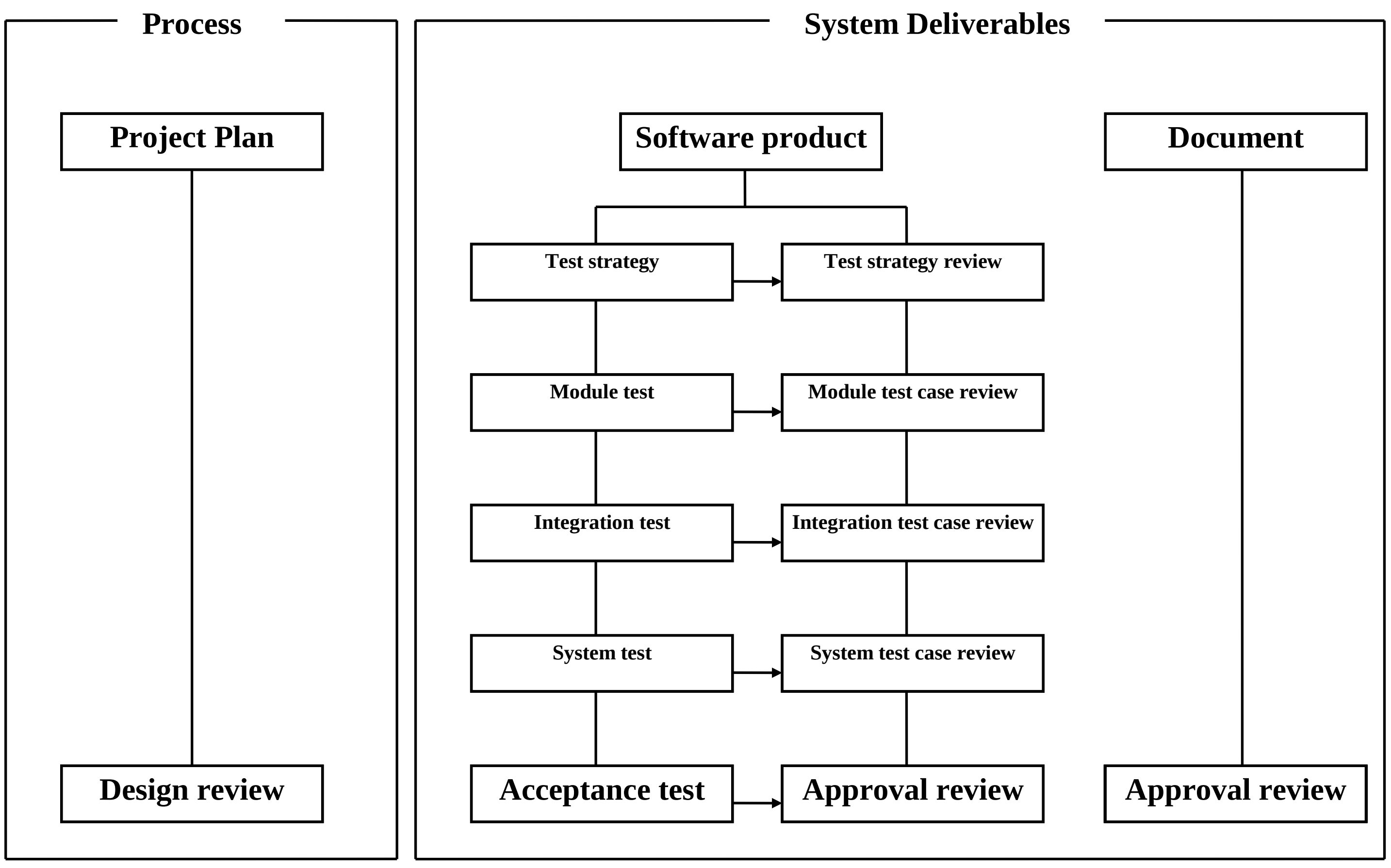


Figure 10. Measurement of results

Measurement of Development Process Control

	Project Planning	System Planning	Testing	Configuration Management	Quality Assurance
Complexity	Total effort 734 mandays	Number of modules 25	Number of test cases 395	Number of Deliverables 17	Number of quality records 1141
Effort used for process	94	59	147	27	83
Effort used for review	21	14	21	2	4
Number of review reports	20	3	10	6	20
Number of non-conformances	4	2	5	0	0
Number of feedback	59	27	37	6	20
Number of feedback reported to project team	37	23	22	4	19
Number of non-conformances recognized after approval	-	1	3	-	-

ISO 9001 Quality System for the organisation

Organisational standards

Number of ISO 9001 compliant quality system documentation	53
Rate of the new standards resulting PIE	85%

ISO 9001 documentation (quality records)

Reviewed period	Apr.-Oct. 1998
Total number of ISO 9001 documentation of the projects	2560 records
Number of the projects	19
Average number of ISO 9001 documentation of the projects	135 records
Total size (effort) of the projects during the reviewed period	1600 mandays
Average size of the projects during the reviewed period	84 mandays/project (6 months)

Maturity tree of org1020/unit1020

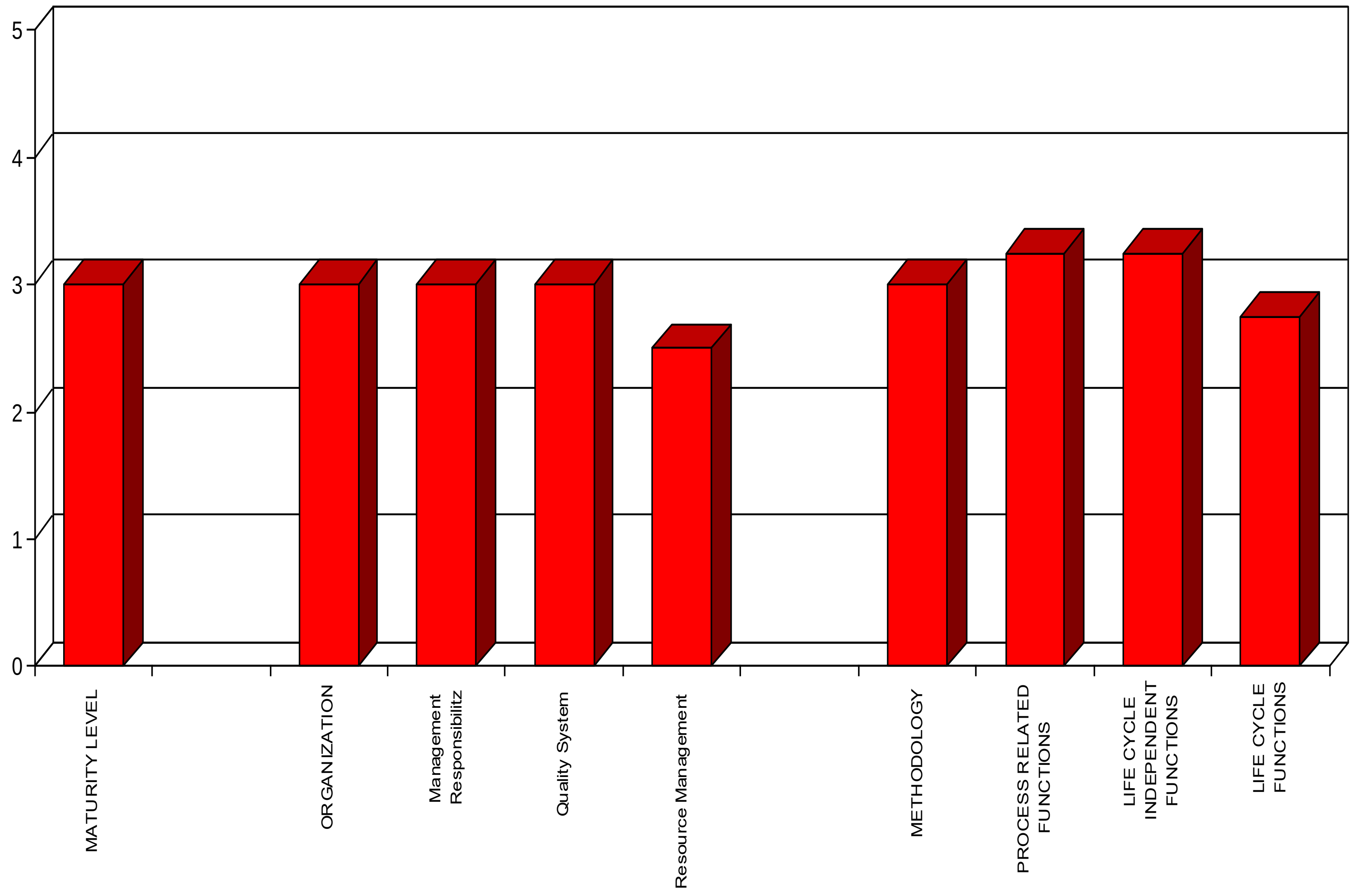
BootCollector	2.31
Algorithm	1.00
Questionnaire	2.3
Date	20-10-98

MATURITY LEVEL **3.00**

ORGANIZATION	**3.00**	**METHODOLOGY**	**3.00**	**TECHNOLOGY**	**B**
Management Responsibility				Existence	
Quality System				Effective use	
Resource Management					

LIFE CYCLE FUNCTIONS	**2.75**	**LIFE CYCLE INDEPENDENT FUNCTIONS**	**3.25**	**PROCESS RELATED FUNCTIONS**	**3.25**
Development Cycle Model		Conf.& Cha. Management		Process Description	
User Requirements		Risk Avoidance & Management.		Process Measurement	
Software Requirements		Project Management.		Process Control	
Architectural Design		Quality Management.			
Detailed Des.& Implementation		Subcontactor Management.			
Testing					
Integration					
Acceptance testing & Transfer					
Operation & Maintenanace					
Special purpose systems					

Profile of main categories
org 1020/unit1020



Session 9 – SPI and Software Life Cycle Support

Enhancing software configuration management for a process control system
23891

Mr (M.SC) Antti Välimäki
Valmet Automation Inc, Tampere, Finland

(PCS) Project Management and Engineering Global Control System

PJ King
Unit 8, IDA Centre, Pearse St, Dublin 2, Ireland
Dr. K Arthur
Unit 8, IDA Centre, Pearse St, Dublin 2, Ireland

Multi-Platform Configuration Management
Process Improvement Experiment

Dr G. Roth
Transaction Software GmbH

Enhancing software configuration management for a process control system 23891

Mr (M.SC) Antti Välimäki
Valmet Automation Inc, Tampere, Finland

.

Introduction

This is the final report of ESSI project 23891- Enhancing software configuration management for a process control system, Confmanag. The objectives of this PIE are to improve software configuration management practices and to introduce measurements as part of our software development process in order to achieve faster delivery of products and improve customer response times, as well as support parallel development, geographically distributed teams and multiplatform environment.
This report describes our experiences about ClearCase and MultiSite tool applications in our baseline project and how these tools have been utilized in achieving our objectives. Furthermore, this report contains our objective related measurement results and the conclusions we have come to about the effects of tools on our working methods and organization.
Many software product developing European companies share similar software configuration management problems and the results of this experiment are useful for these companies. The European Commission through the ESSI/PIE program has supported this project.

Background Information

Objectives

Project objectives are to improve software configuration management methods in software development process and make measurements. The specific objectives are to provide good support

* for parallel development,
* for geographically distributed teams,
* for multiplatform environment and
* for faster customer responses and delivery of products.

One more objective is to measure our maturity level by applying the ISO-SPICE [1] which is being developed within the ISO-framework.

Starting Scenario

Valmet Automation Inc. develops and supplies control and management systems for pulp and paper, chemical and petrochemical industries, as well as power and desulphurization plants. Valmet is the world's leading supplier of distributed control systems for the pulp and paper industry. The distributed control system Damatic XDi is a complex and software intensive product in a demanding market, where high reliability and quality level are the key competitive factors.

Our quality and improvement level

Valmet Automation Inc. received ISO-9000 certification in the year 1992. Ever since we have continued improving the quality of our products.
The next step was to audit our software development process against ISO-SPICE (Software Process Improvement and Capability dEtermination). The audit was held in January 1997 and as a result baseline project’s configuration management [2,3,4] received the capability rating of 3. However, the new requirements will cause our rating to drop, since the current SCCS-based version management do not support the new requirements which are parallel development, geographically decentralised development and multiplatform environment.

Our software development environment and the main problems

The main software development process has been described in our quality system. It in itself consists of several processes, e.g. product development process, project management process and project development support process. Product development process consists of subprocesses: requirement specification, functional specification, design, implementation, module testing, integration testing and system testing. The configuration management is a subprocess of the product development support process and it has links to many subprocesses of the product development.
Rapid prototyping has been applied in development projects. Some pilot projects

have been carried out with object-oriented methodology based on OMT (Object Modelling Technique by Rumbaugh et. al.). Our development environment includes:

- different UNIX-platforms,
- Microsoft Windows 95 and
- Microsoft Windows NT.

Our major development languages are

- C++ and
- C.

The baseline project consisted of porting of the software of Damatic XIS -information system to Windows NT -platform and the development of additional features of XIS. The project was driven by economic needs to keep our competitiveness high.
Our strengths in technical issues were thorough knowledge of UNIX and the functionality of SCCS-based version management in UNIX-platforms. Our weaknesses were the unfamiliarity with new environments and tools.
Our strengths in business were the high quality and usability of UNIX-based XIS-product, and our weakness was the absence of Windows NT –based products, since our customers were interested in Windows NT –based XIS-products – especially when user interfaces were concerned.
One of our strengths in organisation was that our personnel consist mostly of developers with university degrees in software or electrical engineering. Also our experience and knowledge are of high level.

Plans And Expected Outcome

Working Plan

This working plan includes three major activity groups:

1. Starting activities
2. Applying and measurement activities
3. Management and dissemination.

The first phase consisted of starting activities that include ClearCase training courses, the making of instruction documentation and the support for ClearCase users. It started in May 1997 and ended in June 1997. The planned deliverables were the measurement plan and configuration management instructions for the baseline project.

The second phase consisted of applying ClearCase and making measurement activities for software configuration methods. It started in July 1997 and ended in June 1998. The planned deliverables were measurement documents and new versions of configuration management instructions for the baseline project.

Management and dissemination happened during both phases. The planned deliverables were periodic progress reports, mid-term report, final report, consolidated cost statement, other dissemination documents and Spice assessment report.

Expected Outcomes

- The common method for software configuration management makes this process more efficient.
- The quality and productivity will be increased with automated and checked procedures
- Configuration management will increase visibility of changes. The better visibility will raise the quality of the change management. It will also be easier to analyse possible software configuration management problems and fix them more accurately.
- ClearCase Multisite will enable efficient co-operation with geographically distributed project teams. These teams can be either subcontractors or other offices of Valmet Automation Inc.
- Parallel development projects will decrease the throughput time and improve responses for customers' needs.
- Customers will be satisfied because they will get new versions faster for their current release of the product.

Work Performed

Organisation

PIE-project was handled as a project of its own in Valmet's system development. It had its own PIE project manager and method support persons as its members. Baseline project operated in system development as well and it has its own project manager. The members of baseline project included both Valmet's and subcontractors' engineers. PIE project manager arranged e.g. project meetings and other occasions which were participated by both method support persons and, above all, the members of the baseline project. Thus it was ensured that the members of the baseline project were sufficiently supported in introduction of new methods and tools.

Technical Environment

The focus of our PIE was on the evaluation of new methods, procedures, measurements and tools for software confguration management.
Tools used in the project were ClearCase and Multisite for UNIX and Windows NT. Windows NT, HP-UX and Digital UNIX were used as operating systems and Disk Access as NFS software. Before this PIE only HP-UX and Digital UNIX, and of course the SCCS-based homemade version control system, were already familiar to baseline project members.

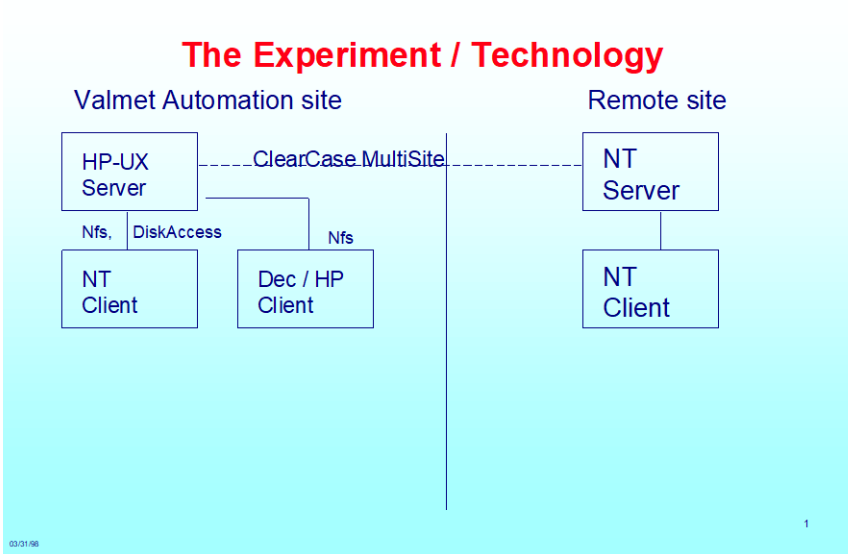


Fig. ATV.1 : The technology used in experiment

Training

A training course of one day was held in the beginning of PIE-project in June 1997 for our ClearCase administrator. ClearCase users (i.e. the members of the baseline project) went through two days' training. In addition to this, one of the members of the method support group, who was the actual ClearCase and Windows NT support person, gave the members of the baseline project support during the PIE-project when needed. This person had familiarised himself thoroughly with ClearCase before the beginning of PIE-project.The basics of ClearCase were easily learned in this training, but of course this was not sufficient, so the support person made method instructions during the summer and held a training course based on these instructions for the members of the baseline project in autumn 1997. The method instructions described how to utilise ClearCase in the baseline project.

Phases of the Experiment

This working plan includes three major activity groups:

1. Starting activities
2. Applying and measurement activities
3. Management and dissemination.

The first phase was realised almost according to the plan and we made the deliverables which were the measurement plan and instruction documents. The installation of ClearCase and MultiSite succeeded well both in Valmet and in its subcontractor.
The second phase consists of applying ClearCase and making measurement activities for software configuration methods.
ClearCase was used contemporaneously with both stages. Because the decisions about configuration management methods, compiler, the structure of makefile and scripts etc. were already made in the early stages of the project, software designing had a good start. The transfer of source code files to ClearCase database went also quite well, since we transferred only the latest version to ClearCase and left the older versions in the SCCS based version controlling system. Most of the problems were naturally caused by new tools the features of which were not yet all that familiar. In spring 1998 ClearCase 3.0 was introduced in Valmet Automation Inc., which remarkably improved the usability of Windows NT. ClearCase 3.2 was introduced in August 1998, which gave a solution to our performance problem by introducing snapshot view –feature. We were now able to compile locally without using the server.

Project management and dissemination occurred during both phases. Project management consisted of project meetings, reports etc. The first public dissemination was held in December 1997 in Tampere University of Technology in Finland (web page: *http://www.cs.tut.fi/configurationmanagement.html*). The second public dissemination was held in January 1998 in Eurex project at Stockholm, Sweden (web page: *http://www.sisu.se/projects/eurex/WorkshopConfig.html*). The third and official Mid Term dissemination were held in April 1998 in Venice, Italy (web page: *http://galileo.iei.pi.cnr.it/AQUIS98/*). The next dissemination action was the creation of the web page of our project (*http://www .valmet.com/automation/essi/ confmana.html*). The final dissemination will be held in EuroSPI'98 which will take place in Gothenburg, Sweden (web page: *http://www.iscn.ie/conferences/iscn98*).

Results and Analysis

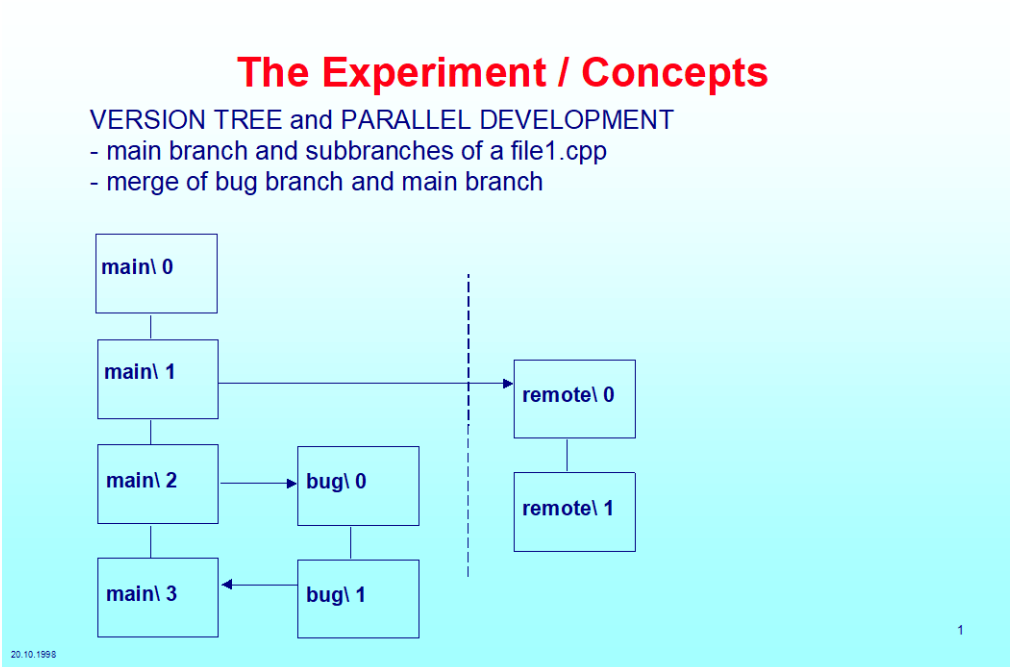
After training courses and instruction making we started the use of ClearCase and Multisite in our baseline project.
At the same time we started taking measurements and comparing current SCCS-based version management with new ClearCase-based management. Measurements were taken from a group of 64 files from which were generated seven libraries. The measurements were made in an environment where UNIX-servers were used. Files were read from their drives through NFS using UNIX-Client when the files were

either in SCCS-files or in ClearCase database.

Technical

GENERAL COMMENTS ABOUT USING CLEARCASE AND MULTISITE
ClearCase and Multisite are good but complicated tools. The software configuration method of one's own is important to efficiently use these tools.

SUPPORT FOR PARALLEL DEVELOPMENT
Parallel development is much more complicated way to develop software than nonparallel development. Especially here you will need good software process method to manage these situations. There is also a possibility to mistakenly change wrong version of file if you choose the wrong parallel branch from ClearCase. The use of branches is the way how ClearCase supports parallel development.

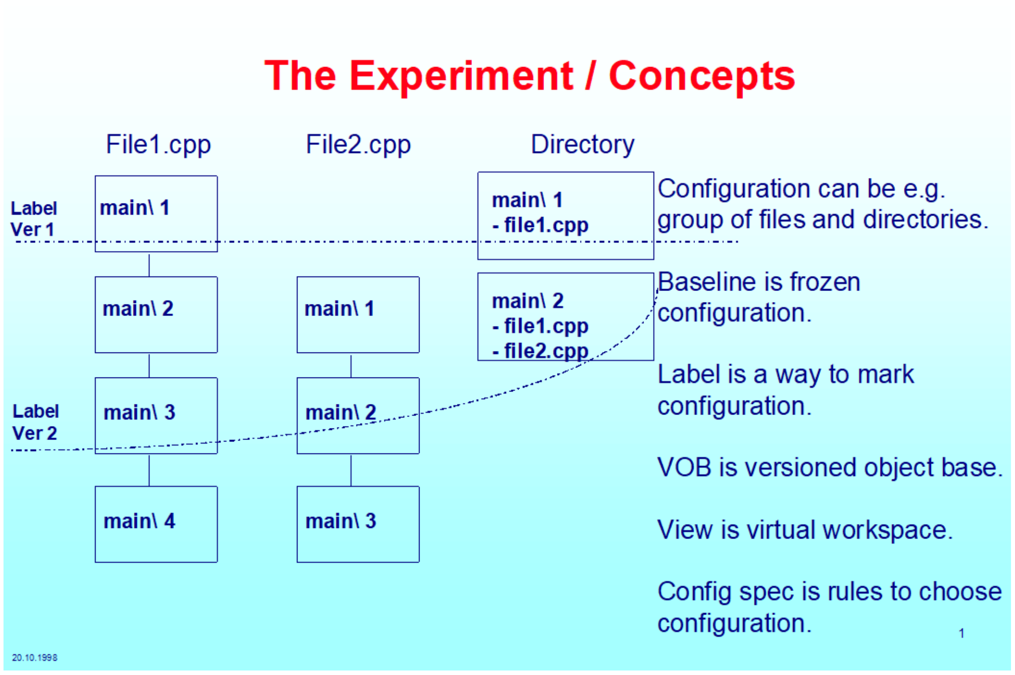

The Experiment / Concepts

VERSION TREE and PARALLEL DEVELOPMENT
- main branch and subbranches of a file1.cpp
- merge of bug branch and main branch

Fig. ATV.2 : Version tree and Parallel development

Version tree is a hierarchical structure in which all the versions of file1.cpp are organized. The arrows show merge operations between different versions of file1.cpp.

Parallel development is a concurrent creation of versions on e.g. two branches in the version tree.



The Experiment / Concepts

Configuration can be e.g. group of files and directories.

Baseline is frozen configuration.

Label is a way to mark configuration.

VOB is versioned object base.

View is virtual workspace.

Config spec is rules to choose configuration.

Fig. ATV.3 : More configuration management and ClearCase concepts.

In the ClearCase parallel development and file merging is supported well. ClearCase also tries to minimise the likelihood that developers would work on the wrong branch by built-in Wizards and View profiles. According to Rational, the process development tool ClearGuide will take it a step further and is able to enforce users to work only on the appropriate branch of the task they are working on.
We also made measurements between SCCS- and ClearCase-based method in parallel development when five different versions of the same files were changed in two different branches and these changed files were combined into one branch. According to this test ClearCase-based method took only one third of the time which was taken with SCCS-based method. To be more specific, it took ClearCase ten minutes to combine the changed files, while SCCS-based method needed thirty minutes for the task.

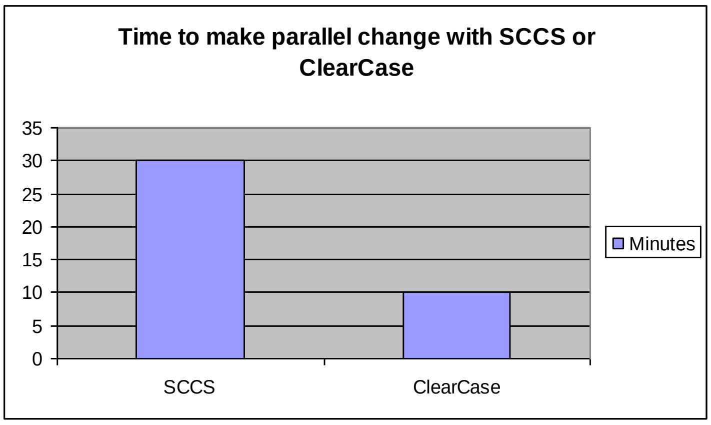


Fig. ATV.4 : Time comparison when making change

SUPPORT FOR GEOGRAPHICALLY DISTRIBUTED TEAMS

Remote development is supported well in Multisite, because Multisite will automatically transfer the changes one has made from one development site to another. Firewalls may cause some problems. One might also experience some errors in sending changed packets. There is a recovery procedure for these kinds of situation, but it is a little bit complex to use. If one wants to read the files which have been developed with Multisite, he needs licenses for both ClearCase and Multisite.
In our measurements we compared the use of Multisite to the use of an ordinary FTP-program (File Transfer Protocol) in transferring changes between two agencies and we came to the following conclusion: In cases when during product development there are three or less subcontractors developing their own part of a system in their own offices, the amount of time and work needed to install and administrate Multisite is comparable to that what is needed to transfer the changed files using an ordinary FTP-program. In all other cases product management becomes so complicated that it is a good investment to acquire Multisite in order to reduce the total amount of work. The latest version (3.2) of ClearCase includes a snapshot feature that is able to utilise the use of an FTP-program described above and reduces the need to acquire Multisite licences for each developer separately.

SUPPORT FOR MULTIPLATFORM ENVIRONMENTS

Multiplatform environment is supported well: one can use e.g. Visual C++ with Windows NT and Emacs with UNIX. Makefiles, however, might become a problem, because they cannot be totally similar in UNIX and Windows NT. There are also many ClearCase properties to ensure the easy use of ClearCase and other tools in a mixed UNIX and Windows NT environment in the same project. This is also one of the ClearCase's strengths when compared to other SCM tools.

PERFORMANCE IN MULTIPLATFORM ENVIRONMENTS

During measurement we have come to conclusion that in UNIX environment compilation of small number of files is equally efficient in both SCCS- and ClearCase-based environments. When hundreds of files are compiled, a ClearCase-based environment requires considerably more efficient UNIX-server (especially because of memory requirements) than an SCCS-based environment.

In Windows NT environment the compilation of files in a local drive was twice as fast as it was in a ClearCase database located in a UNIX-server. Fortunately the latest version (3.2) of ClearCase has snapshot feature which can be used to transfer the files to a local drive to be compiled there instead of a server drive thus considerably improving ClearCase's performance in Windows NT environment.

SUPPORT FOR FASTER CUSTOMER RESPONSES AND DELIVERY OF PRODUCTS

New releases of product can be released fast because of short release build time. The delivery of product releases depends on manufacturer's policy on this process area, but ClearCase gives quite a good support for fast building of new product releases.

When comparing release build times we came to conclusion that ClearCase-based environment was about 30% faster than SCCS-based environment. In our SCCS-based environment old release was searched from an optical drive in which old release has been copied during delivery. In ClearCase based environment it was searched from a database. ClearCase-based solution has some other strengths like easy labelling of delivered release thus making old releases easier to find, and the possibility to easily combine these kinds of corrections with an old release as has been earlier described under item ‘SUPPORT FOR PARALLEL DEVELOPMENT’.

Business

ClearCase and Multisite are good but complicated tools and they require a lot of hardware resources. The licences are also quite expensive. Furthermore, a dedicated support person is needed to help beginners in using these tools and in exceptional cases.
Parallel development support will decrease the throughput time and improve responses for customers’ needs - especially with short release build time of the old releases. This is essentially realised in maintenance phase, when possible error corrections are needed for the customers as soon as possible.
Multisite has enabled a co-operation between geographically distributed project teams. It enables the flexible use of subcontractors from the sites with sufficient technical knowledge.
The final audit of the baseline project was held in May 1998 and as a result baseline project’s configuration management received a capability rating of 3- which is a good achievement and shows that the new requirements have been fulfilled.
Multiplatform environment support has enabled easier and faster offer of Windows NT –based products to our customers. This has been economically remarkable change, since customers’ interest in Windows NT –based products has been considerable.

Organisation

A dedicated ClearCase support person is needed to help beginners in using these tools and in exceptional cases. This support person can support many development projects if needed.
The possibility for parallel development has made the development work more complicated. Therefore one additional person is needed to control the configuration management issues inside the project. The next chart shows the number of support calls ClearCase support person received during the PIE-project:

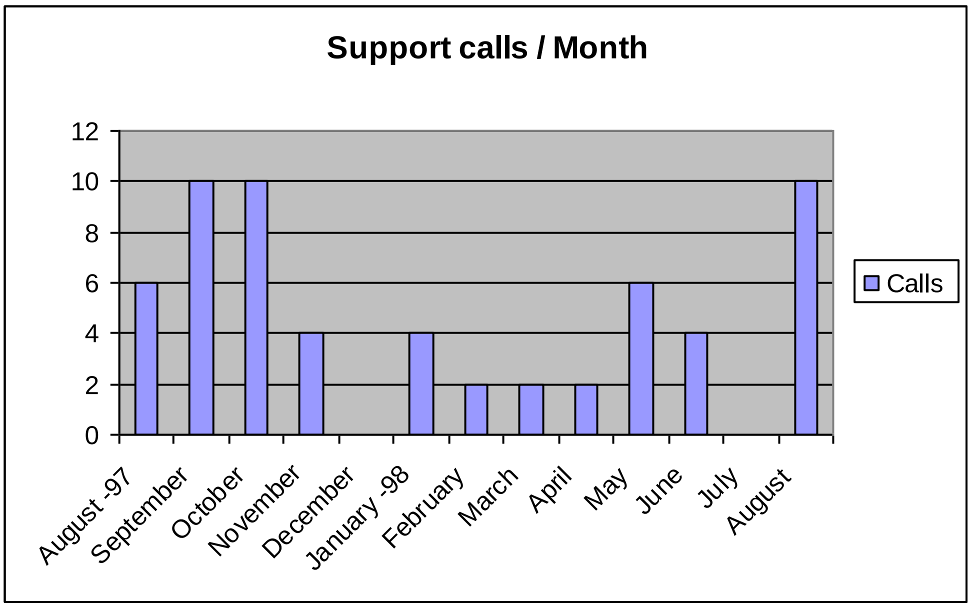


Fig. ATV.5 : Support calls / month

The average amount of work used in support was ½ days / week while the maximum was 2 days / week and the minimum 0 days / week.

In autumn 1997 the greatest cause of increased workload was the introduction of new methods in PIE-project. However, the amount of work spent in support remained reasonable, because we were able to provide PIE-project with good product management instructions and training. In spring 1998 ClearCase was updated to version 3.0 and in August to version 3.2.

What circumstances concerning tools reduce the need for support?
+ if users know how to use the tool correctly
+ if tool recovers from incorrect usage
+ if the rest of the development environment e.g. Windows NT –environment, UNIX-servers and compilers function correctly
+ if other members of the group are able to help if need arises i.e. the general level of knowledge about ClearCase and operating systems is high enough
+ ClearCase support person takes care of the most difficult parts, if it takes too much of actual designer's time to learn them

What circumstances concerning methods reduce the need for support?
+ correctness and exactness of documentation
+ some kind of version control management (e.g. SCCS) is already familiar

What circumstances increase the number of support calls?
+ if support person is competent and polite
+ if support person can find a solution to the problem

Culture

The new methods and tools always meet some resistance from people. Because of this, training courses and good support were needed to help people to learn these new methods when working with ClearCase and Windows NT. Users have, however, understood the importance of new product management requirements and been satisfied with ClearCase since it facilitates working especially in Windows NT environment. The introduction of Windows NT in addition to UNIX as a development environment has required a lot of work as such.

Skills

The members of the baseline project in this PIE gained valuable additional skills as they learnt to use ClearCase in their project.

Key Lessons

This section summarises the key lessons that we have learnt from the process improvement experiences.

Technological Point of View

From a technological point of view we have learnt the following issues:
- ClearCase and Multisite are good but complicated tools for software configuration management.
- Parallel development and file merging are supported well, but specific rules and roles are needed to manage this kind of development.
- Remote development with Multisite is supported well, because Multisite is able to automatically transfer your changes between the differenent development sites.
- Development environment’s change from UNIX to Windows NT is a complicated matter and requires a lot of work.

Economic Point of View

From an economic point of view we have learnt the following issues:
- Parallel development support will decrease the throughput time and improve responses for customers’ needs, especially with short release build time of the old releases. This is essentially realised in maintenance phase, when possible error corrections are needed for the customers as soon as possible.
- ClearCase requires a lot of hardware resources. Developers need licenced versions. Furthermore, a dedicated support person is needed to help beginners in using these tools and in exceptional cases.
- Multisite has enabled a co-operation between geographically distributed project teams. It enables the flexible use of subcontractors from the sites with sufficient technical knowledge.
- Multiplatform environment support has enabled easier and faster offer of Windows NT –based products to our customers. This has been economically

remarkable change, since customers' interest in Windows NT –based products has been considerable.
- Tools and methods are good investments, because if they are utilized efficiently, it reduces time wasted in product management problems thus enabling designers to concentrate on developing new features.

Strengths and Weaknesses of the Experiment

The PIE provided us a chance to use and measure the using of ClearCase and Multisite in a real project. This kind of opportunity itself was valuable for us.
From the organisational perspective, the experiment was a success because ClearCase and Multisite seem to be suitable tools for our other development projects.

One of our strengths was that we started the experiment from proportions supported by Windows NT (e.g. user interface). We also had a good ClearCase support person who had profound knowledge about ClearCase and Windows NT and was thus able to give good support. To mention some other strengths, the members of the baseline project had high level of knowledge and experience and we had other development projects where ClearCase was used as well so we could diversify gained results and their analysis.

Our weakness was, as always in development projects, the lack of time which is the reason why new tools and methods can be tested and utilised step by step.

When looking back now, it seems that we have succeeded in our objectives in this experiment.

Conclusions and Future Actions

As a conclusion we can state that our process improvement experiment was useful. The baseline project consisted of porting certain parts of the software of Damatic XIS -information system to Windows NT -platform and the development of additional features of XIS. This work will continue and we are going to use ClearCase more in our other development projects.

We are also going to try to model our software configuration process by use case method [5] which seems to be a good method to describe rules and roles.

Glossary

C++	A programming language
Confmanag	Enhancing software configuration management for a process control system
PI	Process improvement
PIE	Process improvement experiment
SCM	Software Configuration Management

References

[1] ISO/IEC 15504. SPICE - Software Process Assessment - Part 5: Construction, Selection and Use of Assessment Instruments and Tools, Version 1.00, ISO/IEC Copyright Office, Geneva, Switzerland, 1995, 130 p.

[2] H. Ronald Berlack, Software Configuration Management, John Wiley and Sons, Inc., New York, New York, USA, 1992; ISBN 0-471-53049-2.

[3] Stephen B.Compton and Guy R.Conner, Configuration management for software, Van Nostrand Reinhold, ISBN 0-422-01746-4, 1994.

[4] Fletcher J. Buckley, Implementing Configuration Management: Hardware, Software and Firmware, IEEE Computer Society Press, 1992.

[5] Jacobson I., Christerson M., Johnsson P. and Övergaard G., Object Oriented Software Engineering, A Use Case Driven Approach. Addison-Wesley, 1992.

Appendix 1

Antti Välimäki's CV

Mr Antti Välimäki was born in Jalasjärvi (Finland) in 1961. He graduated as Master of Science from the department of Electrical Engineering in Tampere University of Technology in 1986. Since then Antti Välimäki has continued his postgraduate studies studying software development in Tampere University of Technology

After graduation he has been an employee of Valmet in various positions as follows:
1986 - 1988 software designer in Valmet Process Automation Inc.
1988 - 1990 chief designer in Valmet Process Automation Inc.
1990 - 1994 group manager in Valmet Automation Inc.

During the years mentioned above, Antti Välimäki participated in the development of Damatic XDi Distributed Control System in several tasks including implementation, design, specification, project and configuration management.

Since 1994 he has been as a software development manager in Valmet Automation Inc. In this current position his main responsibility is to develop software development methods and quality system for the whole R&D department.

Appendix 2

Valmet Automation Inc. (VAI)

The first part of VAI is Control Systems which develops and supplies control and management systems for the pulp and paper industry, the chemical and petrochemical industries, the metallurgical industries, and power and desulphurization plants. Valmet is the world's leading supplier of distributed control systems for the pulp and paper industry.

The second part of VAI is Measurements which has designed and manufactured transmitters for the process industry since the early 1950's. The greatest developmental challenge has always been to keep measuring technology in line with the changing needs of process industry.

The third part of VAI is SAGE Systems which is the leading North American supplier of supervisory control and data acquisition (SCADA) systems for the remote monitoring and control of oil and gas pipelines.

The fourth part of VAI is Sensodec which offers you the most advanced technology, proven reliability and a wealth of experience in analyzing your paper making machinery condition, process stability and product quality.

If you want to know more, visit our web page which is www.valmet.com/automation.

(PCS) Project Management and Engineering Global Control System

PJ King
Unit 8, IDA Centre, Pearse St, Dublin 2, Ireland
Dr. K Arthur
Unit 8, IDA Centre, Pearse St, Dublin 2, Ireland

Introduction

Clockworks International specialises in the development and localisation of software and Internet products for the global market. Founded by software engineers and run along the technical and management principles of a software company, Clockworks positions its software engineering culture as its competitive differentiator. Activities include localisation of business software, multimedia systems development and development of consumer software.

Starting scenario

Procedures Analysis

The objective of this experiment was to improve the software configuration management and version control systems used by Clockworks. To do this we had to analyse how Clockworks performed. Before the start of the experiment Clockworks had a relatively informal approach to the software process. With small projects involving only one or two trusted staff this approach had been adequate. However, as Clockworks' client base and work force expanded, it became increasingly difficult to ensure that procedures were observed and that difficulties caused by loss of version control could be avoided. Consequently, we conducted a series of sessions to look at working methods and discuss potential improvements. This was not a complete

software process assessment, but allowed us to begin to address the quality of our software process.
The main requirements that were identified from the analysis of Clockworks' procedures and working methods are listed below:

- A system capable of managing large number of separate items (source code, bitmaps, etc…). Configuration Management tools, such as MS SourceSafe, PVCS or RCE, would be tested to assess if they would meet Clockworks needs.
- Good organisation would allow Clockworks to incorporate new releases of a product while the localisation is in progress.
- Good communication was seen as necessary to allow Clockworks to co-ordinate the localisation activity, which is often carried out by many separate organisations and individuals, frequently at great distance from one another.
- Information about the localisation files involved in a project needed to be recorded so that new staff would have sufficient knowledge about ongoing work. This information would also be useful if a project were delayed and then restarted some months later.
- Faults had to be recorded since it was often difficult to know which faults had been corrected in which versions of products.

Team Structure

The experiment team involved five people with specific responsibilities in the various stages of the PIE. A brief description of the members' skills is given below:

- The Project Manager has a high degree of technical understanding, knowledge of quality issues and good communication skills to maintain the required level of co-ordination between the PIE team and the baseline project team.
- The Production Manager is an expert in localisation projects and highly trained in the procedures and resources that are necessary to carry out a localisation project.
- The Network Administrator is an expert in Clockworks' technology and network structure.
- The Engineers are highly involved in the technical aspects of the project. They are responsible for the application of procedures and the use of technology in the localisation of the different components involved in the project.

Company Context

Clockworks' localisation process is a software process with all of the attendant considerations and problems normally associated with the production of complex systems. However, it is further complicated by several factors. It has been estimated that a typical software localisation project can involve up to 60 separate organisations acting in one capacity or another, whether it be to provide programming skills or project management (as in the case of Clockworks), translation services, art work, or other services.
Clockworks must track the location and progress of every single item that forms part of a software product as it is exchanged between the different project participants. Prior to the commencement of the project, developers were being asked to use a file directory structure and naming convention intended to provide a reasonably

consistent way of storing the different versions of each product that arrived and were despatched for various purposes. The files were stored on a central server and available for read and write access by all project teams. This approach offered no means of documenting which items of a given version had been altered, and which items were currently being worked on by which individual.
Once all of the elements had been re-engineered, each distinct new version of the software was consolidated by Clockworks staff and checked. Any detected defects were logged and corrected. No consistent procedures were used for reporting and tracking defects.
The checked and corrected product versions were then passed to the customer who conducted acceptance testing. Faults were reported by the customer to Clockworks, and these had to be rectified before a new version of the software was returned.
The increase in workload and growth in number of staff members led Clockworks' management to look for more structured and sophisticated ways of organising the work.

Baseline project context

The localisation of the product "Oil Change" from Cybermedia Corporation was the first project where the new configuration management system was tested. The project involved localising the product from US English into German, French, Italian, UK English and Japanese.
Shipments to the client involved approximately 125 small software files each week for a period of six months. The client provided acceptance QA on the files, reporting any defects and returning defective files for rework. The localisation of this product had already started and the error rates appeared to be too high. The failure error rate for the delivery of files was in the 30% to 40% range and the repeat failure error rate was almost 90% (refer to figures from week 24 to week 32 in Appendix C). At this stage the project was suspended in order to formalise the testing procedure and apply the new configuration management procedures and Lotus Notes approach with the hope of improving the quality. The project was resumed in week 34 of 1997.
Metrics to assess the success or failure of the experiment were collected during the project. It provided an ideal scenario to compare figures before and after the new procedures came into effect (refer to Appendix C for a summary of figures from the baseline project).
This project consisted in the localisation of 5000 files and it was one of the larger projects that Clockworks was dealing with when the experiment started. At the moment it would be average in comparison with the size of other Clockworks assignments.
The turnover in the baseline project team was static. The project team consisted always of the same individuals.

Plans and expected outcome

Technical Objectives of the PIE

Given the analysis of the methods then current a set of goals were devised. The technical measurable objectives of this PIE were:

1 To reduce by 50% the incidence of incorrect configuration when versions of software products are released, avoiding the need for rework and thereby increasing customer satisfaction. The error rates before and after the new technology and procedures were measured in the baseline project, allowing us to assess the impact of the new approach in the configuration management of localisation projects.
2 To provide better control over the defect correction process, reducing by 50% instances of reported defects being missed or recurring after having been corrected. Measuring the repeat failure rate allowed us to assess the impact of introducing the new defect correction procedures in the baseline project.
3 To allow control to be exercised over changes to products so that the complete change history of any component of any product version is available, showing who made each change, when it was made and why. The Lotus Notes database system that was developed as part of this experiment stores this and other information.

Business Objectives of the PIE

The specific measurable commercial objectives of this PIE were:
1 To reduce by 30% costs attributable to rework caused by software defects and errors in configuration when software is released to the client (which should have been achievable if configuration management were improved).
2 To reduce by 30% costs of time spent on configuration-related administration.

Other less measurable but important commercial objectives were:

- To lessen the current dependence on specific key members of staff, allowing Clockworks to introduce new staff into the software process more easily by documenting project knowledge as part of the formal software configuration management procedures. It was envisaged that this would allow Clockworks to take on more projects and run larger projects and that the ability to tender for larger pieces of work would provide Clockworks with a strong commercial advantage.
- To strengthen Clockworks' relationships with its clients. Taking more responsibility for correct configuration and detection of defects was expected to boost Clockworks' credibility as it tried for increased business and larger projects.

PIE Implementation

Overview

A critical part of this PIE was in selecting the correct software tools. After meetings were held, project members decided to consider three version control tools for experimentation. PVCS Version Manager, which has management of multiple versions of files and protection of software assets among its main features, was soon rejected due to its incompatibility with the current computer system. For instance developers resisted using its DOS interface as Clockworks uses a Windows NT operating system.
Another tool, MS SourceSafe, was tried on the localisation of Corel Printhouse, a product from Corel Corporation. SourceSafe was preferred to PVCS as closer to current practice and although developers agreed to use it, it was finally discarded due to its low capacity to adapt to the nature of the project and the delays that its use was introducing. A list of the main problems identified while trying this tool in the Corel Printhouse project is given below:

- Library differencing is not implemented. This means that full copies of different versions of each file have to be stored. Consequently, the amount of storage used is too large to be acceptable. At the same time retrieval of documents is very slow.
- Code files that are text can be stored without major problems. On the other hand, retrieval and management of binary files (AVI, BMP, QuickTime,…) is slow and does not always work.
- The tool performs too much redundancy checking, which encourages people to ignore warning messages.
- The file structure is difficult to modify.

Similarly, only some features of RCE (the third tool considered) were found useful. It automates the storage, retrieval, logging and identification of multiple revisions of files and it runs on several platforms (Windows NT and 95, OS/2). However, it was not easily adaptable to the different localisation projects.
In conclusion, we found that established packages allow for little variation from project to project and for little variance in file structure. This clashes with the high diversity of our projects (some of them need a lot of testing, some others have a great diversity of file types, some only involve one file whereas others involve hundreds of different files). Instead the project team decided to implement a more flexible alternative, which uses a Lotus Notes Database to store information about files. This approach places a great emphasis in training individuals, and relies on them to perform procedures correctly.
Clockworks management and staff participated in workshop sessions to develop improved procedures that affect the file management, defect correction and error tracking processes.

Phases of the Experiment

The PIE involved the following steps (Appendix A contains the schedule of the project phases):

1. Preparatory Work and Management.

- WP1 Project Management. During the course of the project ongoing scheduling and progress monitoring ensured that the work in the project was coordinated and

issues resolved.
Deliverables Monthly Progress Summary.

- WP2 Needs Analysis. In this stage Clockworks management and staff identified and documented the outline requirements for improved software configuration management procedures and tools. These requirements were identified by exploring problems and concerns in current methods of operation.
  Deliverables Software Configuration Management Requirements and Selection Criteria (Clockworks Internal Document "Configuration Management"[1]).
- WP3 Tool Selection. Objectives of this phase include examination of available automated software configuration management tools, recommendation on suitable tools, and a decision to purchase one or more. It was considered unlikely that we would be able to find one single tool that would do everything required. In fact we found existing version control tools unsuitable to manage the great number of files and versions and the diversity of files, platforms, languages and applications that are present in localisation projects.
  Deliverables "Recommendations on Software Configuration Management Tools"(Sections 2 and 3 of Clockworks Internal Document "Configuration Management"[1]).
- WP4 Procedure Definition. This phase involved the formulation of specific procedures for software configuration management (and adaptation of other procedures for testing, defect recording, change control, etc., as appropriate). Recommendations were made on reorganised file structure, team structures and developer's roles.
  Deliverables "Software Configuration Management Procedures and Deliverables" (Section 1 of Clockworks Internal Document "Configuration Management"[1] and in "File Management"[2]).
- WP5 Staff Training. Training courses in the procedures identified in Workpackage W4, in the new methodology, and in the new tool, were prepared. The courses include topics such as general localisation issues, Lotus Notes usage, file management and other procedures. Employees directly concerned with the PIE attended these courses but all other employees were free to attend. Also, weekly meetings where held exclusively by participants in the PIE to discuss the experiment progress.
  Great importance has been given to this stage of the PIE. A full-time employee was responsible for the organisation and exposition of these training courses and we have obtained very positive benefits from this practise.
  The course materials were formulated so that they would be useful after the PIE is complete for training of new Clockworks staff.
  To assess the success of the training courses, multiple choice questionnaires were given to the participants. They allowed participants to measure their own knowledge and were used in staff performance reviews.
  Deliverables Methodology Training Course; Procedures Training Course; Lotus Notes Database Training Course; Reusable Training Materials.
- WP6 Mid-Term Review. A comprehensive mid-term review was held involving all PIE participants and Clockworks management. The review assessed the likely impact and risks of adopting the new procedures and the Lotus Notes Database in the baseline project and other projects. The decision was to adopt the new procedures and the Lotus notes approach in the baseline project.
  Deliverables Mid-Term Report; Periodic Progress Report.

2. Application of the Procedures and Tools in the Baseline Project

- WP7 Project Set-Up and Planning. The initial software configuration of the baseline project was recorded using the Lotus Notes Database. The file structure given in Figure KAPJ.1 was customised to conform to this project. The product had to be localised into German, French, Italian, UK English and Japanese and the source program was in US English.
  Deliverables Implemented Configuration Management Tool; Initial Version Configurations (The initial file structure and baseline project files are stored in the baseline project Lotus Notes database).
- WP8 Product Preparation. Preparatory work involved separating source code from text to be translated. The files after preparation were the same for each language. A different set of the same files was sent to translation for each language. All these files were placed in the correct location in the file structure and the status of each of these sets of files was recorded in the Lotus Notes Database and tracked during the process.
  Deliverables Prepared Version Configurations; Initial File Status Report (this information is stored in the baseline project Lotus Notes database).
- WP9 Engineering and Translation. When the various sections were sent out to contractors or assigned internally for work to be carried out, their status was recorded using the software configuration management tool.
  Deliverables Re-engineered Version Configurations; Interim File Status Report (the file structure and file status reports are stored in the baseline project Lotus Notes database).
- WP10 Assembly and Checking. As the re-engineered components of each new version of the product were received, their status and location were recorded using the Lotus Notes Database.
- The QA cycle explained in W4 was applied to the project. Each of the six engineers involved in the project was given a set of files to test that was different from the set of files that the same engineer was given to localise. Faults arising from this testing and the checking following reassembly of the product were logged in the Lotus Notes Database and tracked.
  Deliverables Checked Version Configurations; Final File Status Report; Defect logs (this information is stored in the baseline project Lotus Notes database).
- WP11 Delivery (and subsequent maintenance). The software configuration management tool was used to record the status of software components supplied to the customer, and customer-reported defects were recorded. The PIE monitored defect correction work to ensure that the new procedures and tools were usable and functioned correctly. Any changes to software arising from corrections were recorded using the tool and the status of product versions supplied to the customer for retesting were also recorded.
  Deliverables Final Version Configurations; File Status at Delivery; Defect Logs; Change Control Logs (this information is stored in the baseline project Lotus Notes database).

3. PIE Review and Dissemination Activities

- WP12 Review of Experiment. A comprehensive review was held by PIE participants and Clockworks management to assess the effectiveness of the measures introduced in the PIE. The review team recommended adopting the new

procedures to all Clockworks projects. The file structure has been revised to allow customisation in each different project. This framework maximises the organisation of the projects, reduces management costs, and ensures flexibility in the workplace, leading to greater freedom for the engineer.
Deliverables Final Report; Consolidated Cost Statement

- WP13 Dissemination. During the course of the PIE the results have been and will be communicated to Clockworks staff via a series of seminars.

The first series of seminars was held at the beginning of June'97, at which time the new methodology had been chosen. Configuration tools had been tested, and the new procedures had been defined. This seminar provided a forum which enabled Clockworks to communicate the extent of progress made in the PIE thus far. It provided a clear picture of the objectives of the PIE and of how the desired results were going to be achieved.
The second series of seminars was held once the new methodology, the new procedures, and the new technology had been implemented. An analysis of results was presented in this seminar.
The third series of seminars will be held in the near future, once the effectiveness of the new method has been assessed and a decision made regarding further use of the method. The decision will be made known at the third series of seminars and the possible company-wide changes ensuing from this decision will be dealt with.
A case study report based on the PIE will be developed for external dissemination and a World Wide Web site containing related information set up. A presentation based on the case study will be given at international software process improvement events.

Flexible Database Approach

The approach we have adopted has been a more flexible one using Lotus Notes to store information about the files involved in a project. The following information is stored for each file:

- Version, name and location.
- Who updated the file and when.
- Status of each file

In addition to tracking the progress of each file, an e-mail address is assigned that contains all the correspondence sent and received regarding the particular project. This e-mail address acts as a discussion repository of the progress and issues arising in the project.
The approach of using a Notes database to store information about files does not stop developers being given the responsibility for storing files in the correct location and consequently the possibility remains of mistakes being made and files being lost. To lessen the risk of making such mistakes, a new directory structure and new procedures for storage of files have been introduced. The new file structure is given in Figure 5.1.

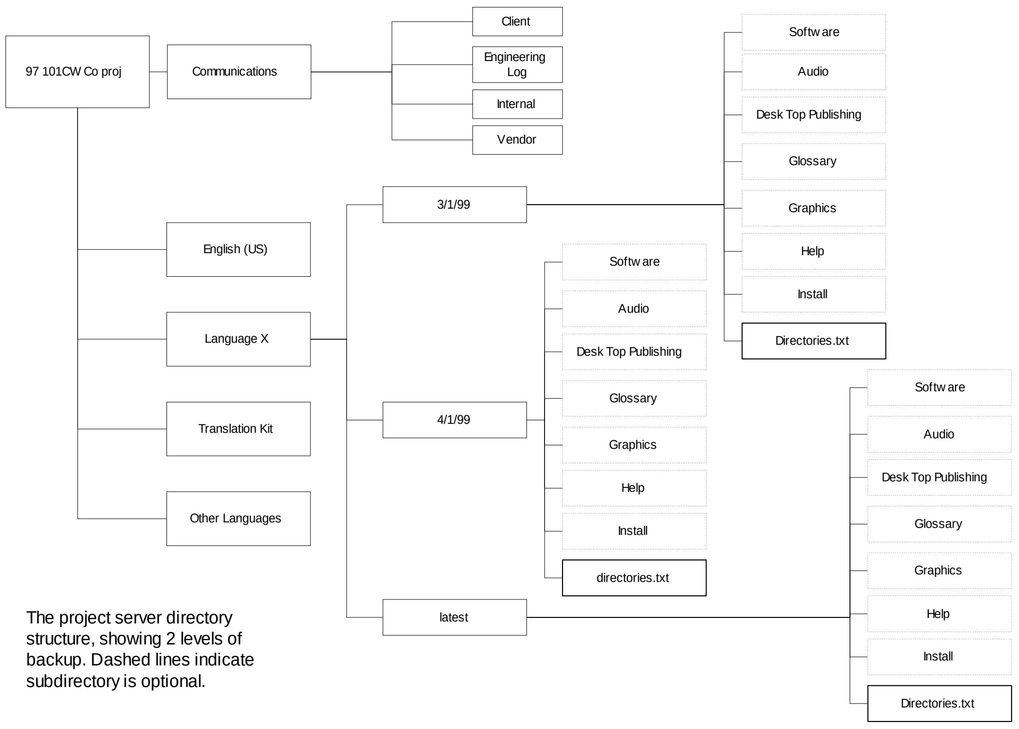


Fig. KAPJ.1 : Directory structure for localisation projects. Ref. [1]

The file storage procedures can be summarised as follows:

- In starting a new project, the production manager assigns a name to the project, which will be the first item of the tree structure above, and the directory structure for the project will be created. The production manager is the only person entitled to name a project and this project name cannot be changed.
- The directory structure will be customised according to the specific project by deleting unnecessary language or component directories.
- Access rights to the directory structure are restricted.
- The project is then assigned to an engineer. The directory structure is copied into the engineer's computer where the pertinent work will be carried out.
- At the end of the day the modified files are automatically copied to the network project directory.
- At the end of a project the lead engineer and the production manager check that all files are properly localised and in the correct locations. When both the lead engineer and the production manager agree that everything is correct, the project files and the change history of each file are archived.

There is also a defect correction procedure in place, which consists of logging the identified defects in the database and keeping track of them. Clockworks' Quality Assurance (QA) cycle helped considerably to minimise the defect rate. Each file goes through three different quality states (Alpha, Beta and Gold). In the Alpha State quality tests are carried out in 15% of the total volume of files. In the Beta State files are moved from the Alpha directory into the Beta directory and new quality tests are performed in 10 to 20% of the volume of files. If the client rejects any file it goes through all three quality stages again. In the Gold State the client has approved the files.

Also, a semi-automated system (a program written in SQA) is being developed that automatically copies the modified files from the engineer's own PC to the Project Server at the end of a working day. This feature eliminates the risk of overriding or deleting useful files.

Plans also include the introduction of Domino servers that will allow data to be published on the Web. This new system will allow our clients to check the state of their products being localised and give them direct access to project budgeting data and other project information. Most importantly it will allow external translators to download files to be translated and then upload them in the right location when they have finished working with them.

In the localisation of 'Oil Change', the baseline project, the file structure that was used was different from the structure given in Figure 5.1. The structure in Figure 5.1 is the final refined structure. In the localisation of the baseline project a customisation of a more general structure outlined in [1] was used (see Figure KAPJ.2). The initial software configuration of the baseline project was recorded using the Lotus Notes Database. The product had to be localised into German, French, Italian, UK English and Japanese and the source program was in US English. The file structure under the directory 'Language X' in Figure 5.2 was replicated for each of these languages.

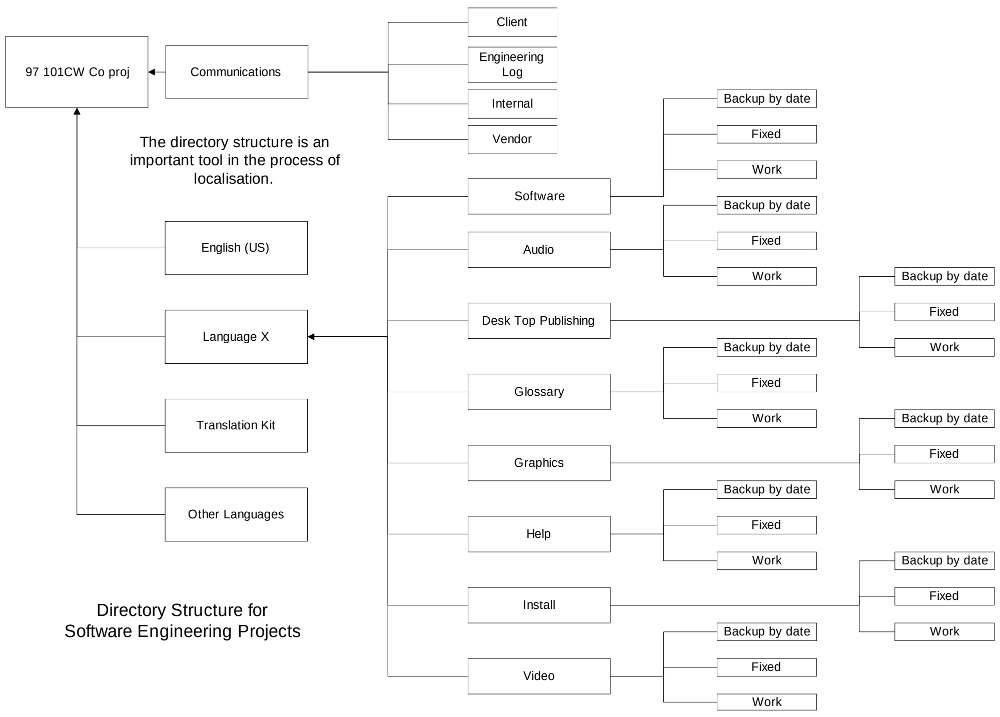


Fig. KAPJ.2 : Customised file structure Ref [2]

Measured Results

Drastic Improvement

The localisation of ‘Oil Change’ required Clockworks to deliver approximately 125 small files every week for a period of six months. The improvement after the new configuration tool and procedures were introduced is remarkable. The failure rate dropped from an average of 35% to an average of 5%. The best results were found in the repeat failure rate that dropped from an average of 90% to almost zero. See Fig. KAPJ.3
At the end of the project the production manager and the lead engineer checked that the files were properly localised. They also reviewed all e-mails received during the project and the lead engineer produced a report stating the main issues that had arisen in the localisation of ‘Oil Change’. This report was stored in the engineering log and the file structure and Lotus Notes database were archived after the project was signed off by the project manager and the production manager.

The following list summarises the final technical results of this PIE:

- The incidence of incorrect configuration was reduced by 85% with the introduction of the Lotus Notes-based system and the new procedures.
- The complexity of localisation projects makes it difficult to use one of the existing configuration tools. Instead, a more flexible approach had to be implemented. A Lotus Notes system is used to track changes and the status of files and a new directory structure has been put in place. These two measures make Clockworks’ software system organised and reliable.
- A defect correction system was implemented to be able to track defects once they have been reported by the client. This system avoided the need to report defects repeatedly. In the baseline project, the instances of errors being reported after delivery was reduced by 90% with the introduction of the defect correction and tracking procedures.
- The new system makes available the complete change history of any component of any product version, showing who made each change, when it was made and why.
- The new procedures improve the archiving and general structure of files.
- The information stored in the database allows us to analyse results to identify possible causes of errors (e.g. individuals not following the procedures right).
- A benefit of this approach is the fact that the solution is not specific to the baseline project and consequently will allow us to reuse the findings in other localisation projects. The database approach is expected to be most useful in larger and longer projects. In small, quick projects it will be far less useful, due to the fact that small projects need less organisation and monitoring.

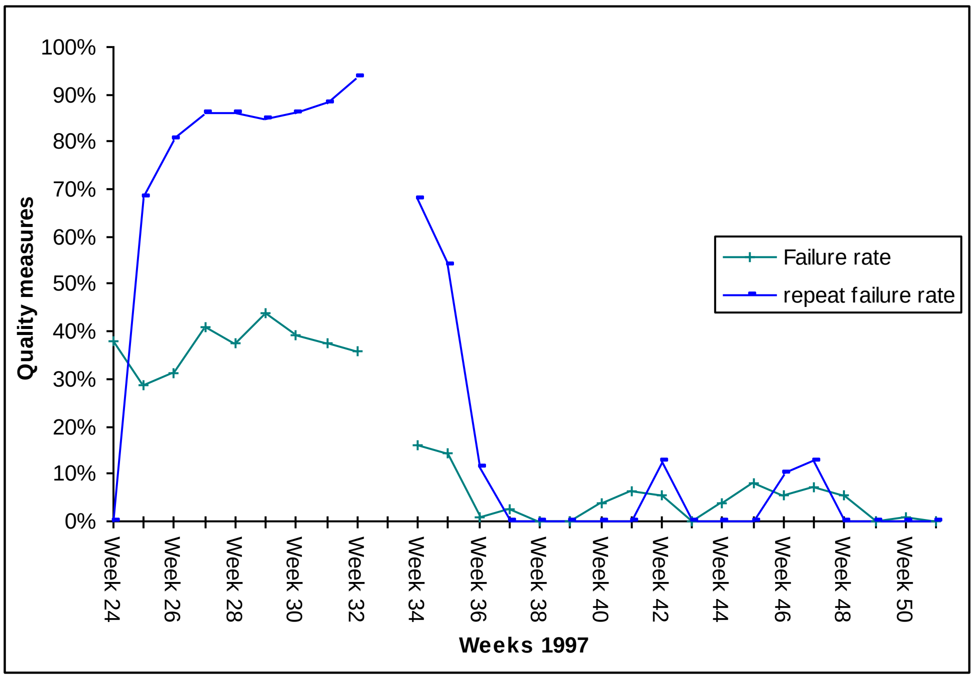
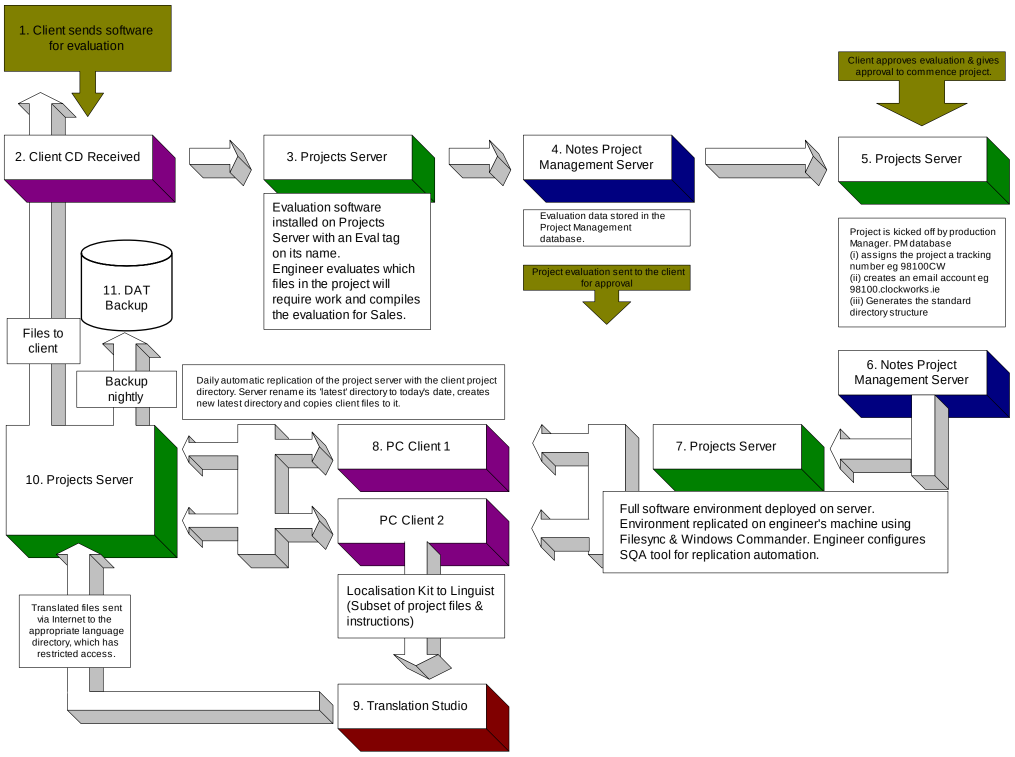


Fig. KAPJ.3 : Failure rate versus time showing the dramatic results Ref [5]



Client approves evaluation & gives approval to commence project.

Evaluation software installed on Projects Server with an Eval tag on its name.
Engineer evaluates which files in the project will require work and compiles the evaluation for Sales.

Evaluation data stored in the Project Management database.

Project evaluation sent to the client for approval

Project is kicked off by production Manager. PM database
(i) assigns the project a tracking number eg 98100CW
(ii) creates an email account eg 98100.clockworks.ie
(iii) Generates the standard directory structure

Daily automatic replication of the project server with the client project directory. Server rename its 'latest' directory to today's date, creates new latest directory and copies client files to it.

Full software environment deployed on server. Environment replicated on engineer's machine using Filesync & Windows Commander. Engineer configures SQA tool for replication automation.

Translated files sent via Internet to the appropriate language directory, which has restricted access.

Fig. KAPJ.4 : Process flow of files from US to localised Ref [8]

Business impact

Some of the final business benefits brought about by the new configuration management policy are listed below:

- The new configuration management system improves Clockworks' file management and general organisation. This allows new staff members to be introduced easily to ongoing projects.
- The reduction in the error rate led to a decrease in the costs of rework and increased customer satisfaction.
- Improved organisation led to an increase in the level of customer satisfaction.
- Clockworks' productivity has improved considerably from the start of this experiment. Training staff members has been a very worthwhile experience and has been one of the major causes of our productivity growth.
- Clockworks' work force has increased to meet the increase in business growth. Currently, Clockworks team consists of 50 employees (at the beginning of the project it consisted of 12 employees). We believe that this business growth is due in large part to the improved customer satisfaction brought about by the new procedures and technology.
- The cost of rework caused by software defects and errors in configuration was reduced by 69% in the baseline project after the introduction of the Lotus Notes database and new procedures.
- The costs related to project configuration and administration were reduced by 52% with the introduction of the Lotus Notes database.

Organisation impact

Participants in the experiment had to learn to use the new Lotus Notes Database and follow the new procedures. They had to record in the database changes made to each file. The network administrator was the only person that could move files to the project server at the end of the day.
Clockworks' workload and work force have increased considerably during the duration of this PIE. At the beginning of this experiment Clockworks employed 12 employees, currently our work force consists of a team of 50 employees.

Culture impact

People involved in the project were unwilling to use any of the existing version control tools. The tools slowed down developers' work and could not hold the great number of files present in localisation projects. However, the organisation was aware of the need for new procedures to support the growth in staff and productivity and people involved in the project presented no resistance to the new procedures and technology. Seminars and training courses made clear to people the objectives and usage of the new system. The new, more structured file management procedure makes work easier for the people using it. Files are easier to track and organisation in general has improved.
The approach used in the localisation of a project before the experiment started was quite informal. The new procedures and the Lotus Notes approach gave structure and guidance to the engineers to the extent that they would not work without them.

Skills impact

People involved in the experiment had to become familiar with the new directory structure and with the new storage procedures. They also had to learn to use the Lotus database to record any progress or change.

Key Lessons learned

Technological point of view

The main lessons learnt from a technological point of view are:

- The diverse nature of localisation projects requires a flexible tool. Existing configuration management tools are too inflexible. The system that has been developed in this experiment is very flexible and cost-effective.
- We believe that improving the organisation of the projects has been a major benefit. The new file structure and notation and the new procedures reduce error rates.
- The new system allows us to keep track of file change history and status. This allows us to reuse this information when a new version of the same product has to be localised. It also allows us to perform data analysis and to identify causes of errors.

Business point of view

Some of the lessons learnt from a business point of view are the following:

- Training people in the new procedures has been an important part of this PIE. Training will be an ongoing task in Clockworks as procedures evolve.
- The solution to our configuration management problems is not reliance on a tool. The responsibility still lies with people who are trained to follow the new procedures.
- Improved organisation increases customer satisfaction because error rates are reduced. Credibility has also improved which is reflected in the increase of business growth that Clockworks has experienced in the past year.
- There are important hidden costs in rework. For instance, it has effects of stress and pressure on staff members reducing productivity and it causes the customer to turn to the competition in future projects.

Strengths and weaknesses of the experiment

The experiment stages were accurately planned and they have been performed as expected. In some cases previous steps had to be repeated or changed in the light of new findings and ideas. For instance the fact that we could not use any of the existing configuration management tools forced us to look for alternative solutions that focused strongly on the development of new file management and storage procedures.

The fact that a production project was used to test the experiment results proved that the new system was capable of being used in real settings. At the same time, this fact presented some disadvantages, for instance, due to the complexity and severe time pressure present in localisation projects metrics were not easy to collect in some cases.

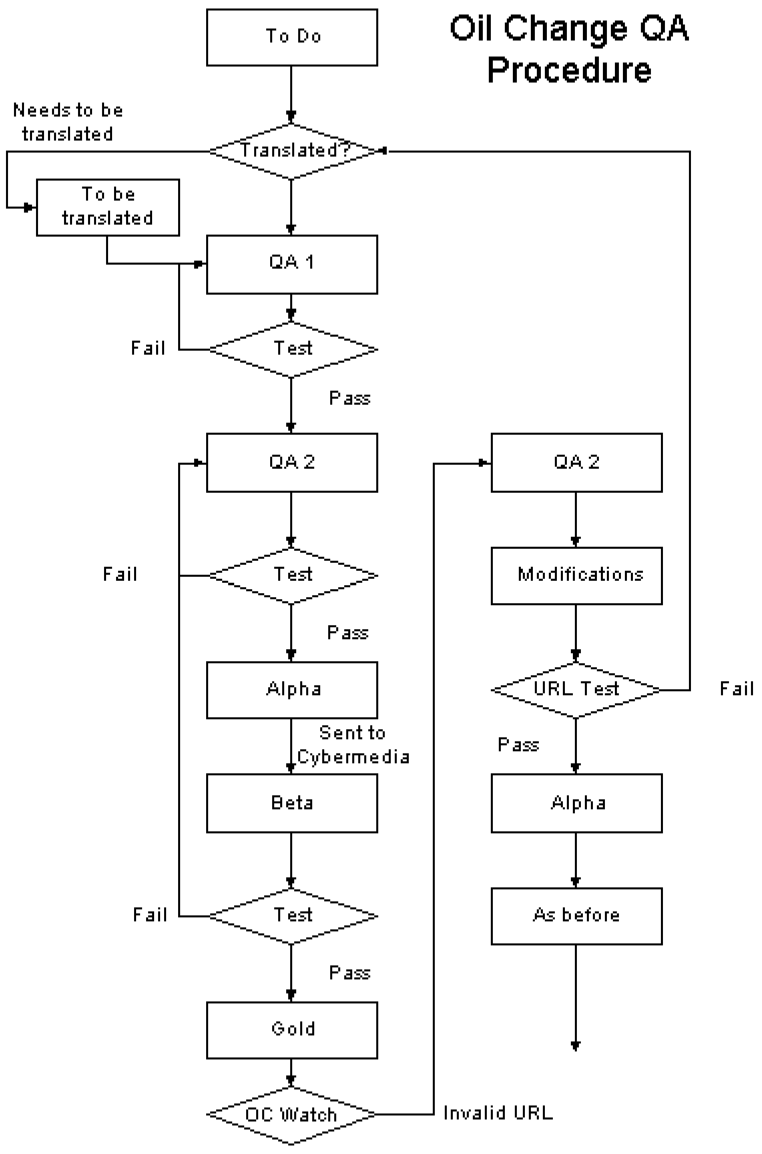


Fig. KAPJ.5 : QA process through which Oil Change files go. Ref [5].

Conclusions

The new procedures and technology were tested first in the localisation of "Oil

Change”, a product from Cybermedia Corporation. However, the new way of working has been designed with the intention of applying it to other future localisation projects and Clockworks intends to reuse it in all its projects. For each new project the Quality Assurance policy will be agreed with the customer and then the Lotus Notes database and directory structure will be customised to satisfy specific needs.

The final assessment of the experiment is that is was satisfactory. We believe that the solution that has been developed in this PIE is highly beneficial for Clockworks and it will provide even more competitive advantage when the new technology and procedures are applied to all Clockworks projects.

We believe that the new technology and procedures are the best option available to the specific needs of our organisation. The preparation stage of the experiment and accurate examination of requirements was a major step which has allowed us to come up with a flexible solution that can be applied to serve a large diversity of projects.

From a technical point of view we plan to extend the functionality of the Lotus Notes database. We also intend to introduce Domino servers that will allow file status and project related data to be published on the Web so that our customers and translators can access it directly. This new Web part of the system will improve communication between the various companies that are involved in a localisation process and will reduce management and file tracking effort. From an organisational point of view we will continue training people in the new procedures.

References

[1] “Configuration Management”, Clockworks Internal Document.
[2] “File Management” Clockworks Internal Document.
[3] “Configuration Management” Clockworks Internal Document.
[4] “File Management” Clockworks Internal Document.
[5] “Cybermedia QA cycle document” Clockworks Internal Document.
[6] “Oil Change QA cycle” Clockworks Internal Document.
[7] “Project Management server”. Clockworks Internal Document.

Appendix 1

PJ King is Managing Director of Clockworks International. PJ set the company up in 1994 and since then has seen a doubling of staff and more importantly turnover each year since. Currently Clockworks employs fifty people in Dublin and is currently opening an office on the west coast of the United States.

Dr. Kieran Arthur is a former lecturer in Tallaght RTC and Mary Immaculate College Limerick, where he taught mathematics and computer programming. He received his MSc and PhD degrees in Applied Mathematics from Dublin City University, where he worked on problems in non-linear optics and superconductivity.
Kieran joined Clockworks International to set up a training program that takes engineers with a strong background in PC skills and turns them into localisation engineers. When this training program was complete and running smoothly, he approached the challenge of developing training programs for all of the other platforms which Clockworks requires.
He maintains research interests in numerical methods.

Appendix 2 Clockworks International

Clockworks International specialises in the development and localisation of software and Internet products for the global market. Founded by software engineers and run along the technical and management principles of a software company, Clockworks positions its software engineering culture as its competitive differentiator.
Clockworks services allows, our clients to bring their products and services to a world wide audience in the most efficient, cost effective manner and in the shortest possible time frame. To date the company has experienced 100% repeat business.

Services

Our core business activities are software and multimedia localisation. Clockworks is one of the few companies that can provide a total localisation solution to its customers, this encompasses:

- Translation
- DTP
- Graphics
- 2D / 3D graphic animation
- Project management
- Internationalisation
- Audio production and post production (Clockworks has its own in house studio)
- Software engineering - cross platform
- QA / Testing

In keeping with Clockworks development strength, the company has invested heavily in developing competencies in localisation of the UNIX, Aix, Solaris and AS400 platforms.

Clockworks Executive Engineering

This area of our business focuses on on-site Engineering, consulting on internationalisation and localisation management support.

Clockworks Internet Services Centre

Clockworks Internet Service Centre is an new development which facilitates the localisation of Web based products. It covers Web localisation, as well as issues concerning site mirroring and information maintenance.

The Clockworks Approach

Clockworks International is seen as a software localisation company with a difference. As a company founded by software engineers we are in a position to provide our customers with a high quality, highly efficient localisation service.
As a highly focused and respected localisation company, working with leading international software companies, we are renowned for being experts in our own field.

Multi-Platform Configuration Management
Process Improvement Experiment

Abstract

The Process Improvement Experiment (PIE) *'Multi Platform Configuration Management'* is focused especially on the situation and problems in a multi-platform multi-version software development environment which is typical for our company.

The goal of this PIE was to reduce the overall costs of the development and maintenance process, to improve code stability and to achieve higher security during the porting and maintenance phases.

TransAction Software is the developer of the relational database system *TransBase* which runs on many different platforms ranging from Mac to main frame VMS or CDC.
About twice a year new main software releases are generated; in addition some customer specific versions have to be maintained and administered.
We have one source code for all platforms which over the time lead to many compile-switches and made the source code hard to read; in addition, many negligible warnings were produced on the different platforms, which made it hard to detect the fundamental ones.
As source code code control system the rudimentary UNIX-based SCCS was used.

The deficiencies mentioned above had significant impact on the porting efforts to the supported platforms which were about 50% per platform of the original development costs.

Our main targets and activities were:

- Design, establish and introduce multi-platform oriented programming conventions to increase platform independency and code readability especially by eliminating platform specific switches from the source code. By the usage and strict obedience of these conventions we can deploy more and stricter code checkers. This enables us to detect and eliminate platform and porting specific problems in an earlier development phase.
- Improve our multi-version source code control, configuration and maintenance management. We hereby have replaced SCCS by the system MKS which also supports different development releases, paths and so-called sandboxes.
- Improve and formalize our procedures and processes for porting and multi-platform validation. In addition, a formal problem and error tracking facility has been introduced.

To implement and evaluate our approach, our new database recovery component has been chosen as a baseline project. It amounts to about 1500 lines of code which is ca. 3% of the total TransBase code size.

The experiences gained so far are very encouraging. As we had expected, the development costs on the development platform increased. The ratio of porting to development efforts however decreased to about 34%. The reduction of maintenance efforts due to improvements of code quality and portability can be assessed only after a longer period of observation.
The use of the more sophisticated source code control system increased both the security and the concurrency of our development.

Thanks to the positive experience we plan to extend these procedures to all other TransBase modules.

Dr. Roth, TransAction Software GmbH, D-81739-Munich Phone: +49/89/62709-0

Session 10 – SPI & Systems Development Part II

Defect Prevention Techniques for High Quality and Reduced Cycle Time

An ESSI Process Improvement Experiment (PIE)

Abraham Peled, Leeba Salzman, Abraham Danon, Paul Rogoway

Motorola Communications Israel Ltd. Tel Aviv, Israel

Practical Implementation of a CleanBase

Malgorzata Warne

Ericsson Radio Systems , Sweden

Reuse of software development experience at Telenor Telecom Software

Magne Jørgensen, magne.jorgensen@ifi.uio.no, Dag Sjøberg, Dag.Sjoberg@ifi.uio.no, University of Oslo, Reidar Conradi

The Norwegian University of Science and Technology, Reidar.Conradi@idi.ntnu.no

Defect Prevention Techniques for High Quality and Reduced Cycle Time

An ESSI Process Improvement Experiment (PIE)

Abraham Peled
Leeba Salzman
Abraham Danon
Paul Rogoway

Motorola Communications Israel Ltd.
Tel Aviv, Israel

Background

Motorola Communications Israel Ltd. (MCIL) is an industrial company which develops and manufactures radio equipment and radio related products. MCIL's Development Division is one of Motorola's largest development group outside of the United States, and is involved in many main-line development projects of Motorola's Communication Enterprise (CE).

Motorola's products and systems are becoming software intensive at a remarkable rate. In the last five years, the amount of software in MCIL's products has increased dramatically and the number of software engineers has more than doubled. More and more engineering assets are becoming software intensive, and the company estimates that in the coming years, the software content in its products will grow to 90%, and software will become increasingly vital to its competitiveness and success.

In order to meet the increased market demands in functionality of new products, MCIL needs to constantly apply more engineering resources in the development process.

The company has invested heavily in quality, and is eager and committed to preserve this important asset. With new, sophisticated and complex products and systems, it is becoming increasingly difficult to maintain higher levels of quality while reducing the development cycle time, to “rush” to market.

Project Description

The “Defect Prevention Techniques” Process Improvement Experiment (PIE) was established to investigate patterns of defects that commonly occur in the Software Development Process. The Goal of the PIE was to define and implement techniques to reduce the total number of defects in the Development Life-cycle and to prevent certain classes of defects from recurring. [1]

It is well known that correcting defects in later stages of the Development Life-cycle is more complicated, expensive and time-consuming than correcting them in earlier stages. Preferable even to early detection, is the avoidance of defects altogether - the Defect Prevention method. Defect Prevention is therefore the best way to optimize the development process costs and to shorten the development cycle time. [2]

The objective of the experiment was to define and implement Defect Prevention methods and techniques, for the various phases of the Development Life-cycle, to reduce the quantity of defects and then to determine a Strategy for decreasing the Testing effort needed for development projects.

The PIE has produced a better understanding of common defect types, suggested and implemented solutions to avoid them, and established a mechanism to investigate new / remaining defects with the goal of eliminating them.

Starting Scenario

MCIL is a well-established Level 3 organization, with a phased development process, institutionalized formal reviews, automated defect tracking, and a long history of pioneering state-of-the-art software tools and technologies.

In order to initiate the PIE, we selected a project from our Digital Radio development group as the baseline project. The project selected is TETRA - Trans European Trunked RAdio - a new all-European digital radio system which integrates, in a single subscriber unit, Cellular and Dispatch (two-way) communication, as well as short paging messages.

The project has more than one release, and we used one for determining the reference line and a consecutive one to measure the improvements effected by the PIE. Selecting a multi-generation project, with two releases, rather than two distinct projects, helped neutralize the effects of several important variables, which could have otherwise distorted the findings. Thus we ensured commonality with respect to development environment, application domain, work culture, working methods, engineering team and tools.

Plans and Expected Outcome

The PIE plan was comprised of several steps :

1. Create a reference line of root causes based on defects recorded in the initial project release.

2. Based on the profile of common defects, define techniques to prevent specific problems in each phase of development.

3. Implement and disseminate techniques to engineering team at phase kickoffs.

4. During subsequent project development, support modified software development process (see Figure 1) and ongoing causal analysis of new defects detected.

5. Review root causes, analyze changes in defect trends. Evaluate efficacy of new techniques. Determine strategy to reduce test effort while focusing on the most error-prone areas.

6. Disseminate findings and recommend changes to the OSSP (Organizational Standard Software Process).

The existing CMM Level 3 development process was modified to include Defect Prevention activities. A first-stage causal analysis was added to the defect closing procedure, whereby the engineer handling the error/defect would fill out a new Analysis form. This Analysis form which includes Beizer Taxonomy classification, cause category, root cause analysis, containment method and suggestion, is physically attached to the problem report and remains part of the database.

A second-stage causal analysis was added to verify the correctness of the first stage and identify trends and extreme cases which require attention. Phase kickoffs were added to the process to educate the engineers on the common errors of the phase and on causal analysis techniques.

Based on the trends identified in the second stage of analysis, defect prevention techniques and strategies were recommended and implemented. This activity was performed by the PIE team and managed as a separate process.

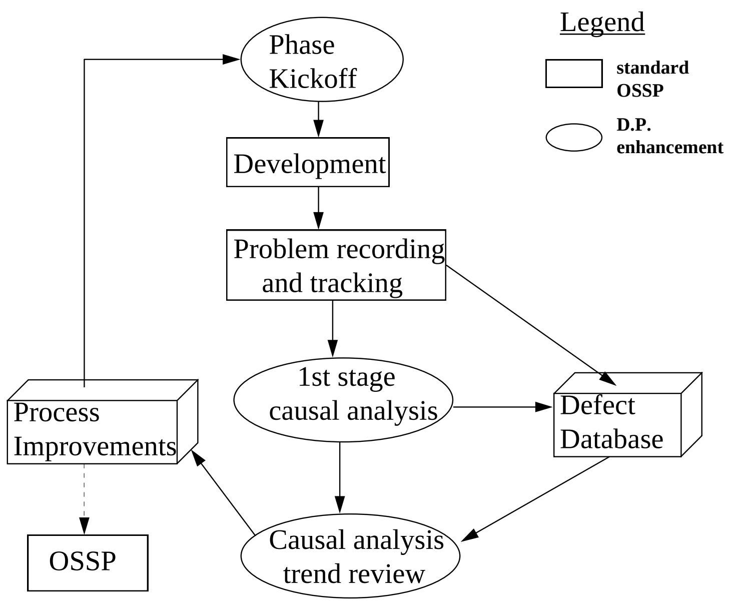


Figure 1 - CMM Level 3 Software Process enhanced for Defect Prevention

The Reference Line : The first step was to perform "Defect Analysis of Past Projects" in order to create a reference line for the PIE. We analysed 1336 defects from the baseline project (TETRA Release 1) and two other projects (to increase the statistical significance). Detailed Root Cause Analysis was performed on all the defects, and the Beizer [3] Taxonomy was used as the "classification vehicle". Analysis was done for five of the development phases, namely : Requirement Specifications, Architectural Design, Detailed Design, Coding and System Test Case Preparation. Based on this analysis, specific Defect Prevention solutions were determined for each of these phases.

The Beizer Taxonomy used for the classification includes ten major categories, each of which is divided into three levels, resulting in a 4-digit number which specifies unique defects. The ten top level categories are :

0xxx	Planning
1xxx	Requirements and Features
2xxx	Functionality as Implemented
3xxx	Structural Bugs
4xxx	Data
5xxx	Implementation
6xxx	Integration
7xxx	Real-Time and Operating System
8xxx	Test Definition or Execution Bugs
9xxx	Other

The causes of the defects as determined by the engineers doing the classification, fall into four major categories:

* Communication
* Education
* Oversight
* Transcription

In creating the reference line, detailed interviews with 24 software engineers took place, in order to fully understand the reason for each defect, to classify the cause and to understand how the defect could have been prevented. This "data mining" was performed on all the defects, resulting in a series of "classification tables" and a good Pareto analysis of the most common problems.

The following Pareto represents the breakdown (in descending order) of the defect analysis according to the Beizer Taxonomy top level categories :

Requirements and Features	(1xxx)	47.0%
Functionality as Implemented	(2xxx)	13.5%
Structural Bugs	(3xxx)	9.3%
Implementation	(5xxx)	8.3%
Data	(4xxx)	6.9%
Integration	(6xxx)	5.7%
Real time and Operating system	(7xxx)	4.9%
Test definition or Execution bug	(8xxx)	4.3%

Within each development phase in the baseline project, we further classified the defects, based on the Beizer Taxonomy. For example, in the Requirement Specifications Phase, the second level breakdown of the main defects was as follows:

Requirement Completeness	(13xx)	37.5%
Requirement Presentation	(15xx)	34.7%
Requirement Changes	(16xx)	11.2%
Requirement Incorrect	(11xx)	8.7%

The third level breakdown of the main "Requirement Completeness" defects was :

Incomplete Requirements	(131x)	73.4%
Missing, unspecified requirements	(132x)	11.2%
Overly generalised requirements	(134x)	4.6%

The same type of data analysis was performed for each of the development phases selected for the PIE.

The next step was to identify a tool-set of phase-specific improvement activities, based on the root cause analysis, that would prevent the defects from recurring in the next release. Highest priority was given to the most common defect types.

Extensive training and phase kickoff meetings were held to empower the development team to integrate Defect Prevention into the existing process. The improvement activities determined in the analysis phase were then applied by the development team in the different development phases, and ongoing defect recording and measurements were performed.

The final step was to compare the numbers and types of TETRA Release 2 defects with those of the reference line. The effectiveness of the "prevention tool-set" was measured in the quantity and types of defects found in the second release of the project. The prevention actions which were found to be effective could then be integrated into the OSSP to improve quality and cycle time for all the projects in MCIL. The impact on the OSSP, including changes to Review Guidelines and changes to the Phase Kickoffs, are considered to be part of the PIE results.

The Expected Outcomes :

1. A framework for establishing a Defect Prevention program in a software development environment

2. A list of improvement actions to be taken by the TETRA project development group in order to prevent defects, including :

   * Method to number the Requirements in the SRS document
   * A Writing Strategy procedure to reduce the ambiguities in the Requirement Specification phase
   * A utility to support/implement the Writing Strategy
   * Improved Software Requirement Specifications (SRS) template
   * Formalised Context Diagram / Feature Interface Chart for the Requirement and Design phases
   * Improved Review Checklists for all phases of the development life-cycle
   * Causal Analysis procedures and meeting guidelines
   * Improved Kickoff meeting templates and guidelines, for all phases of the development process
   * Testing Strategy

3. Improved quality of the Tetra product, including :

   * Decrease in the overall number of defects found in the various development phases
   * Shift in the distribution of defects, by phases
   * Lower development costs
   * Shorter cycle time

Implementation of the Improvement Actions

Kickoff meetings were held for each phase, where the importance of Defect Prevention and causal analysis were explained and emphasised. The improvement actions for the specific phase were presented and discussed. The actions, as suggested by the PIE team, were generally well received by the TETRA development engineers and managers. Techniques such as improved review checklists were applied immediately after the kickoff at formal peer reviews.

In each progressive phase, engineers became more adept at recording the defects using DDTs® - Distributed Defect Tracking System, and at performing causal analysis. They became more open minded about reporting and recording their own defects, understanding the importance of a systematic tracking approach to the quality of the product and the process.

Kommentar [PA1]:

Many TETRA engineers expressed satisfaction with the causal analysis process and kickoff meetings, which made them feel better equipped to prevent defects, and improved their general attitude towards the software process.

The PIE is considered by the technical staff as well as the business staff, to be a positive process, which gives us an advantage in better quality of the products, and reduced cycle time of the development process. As such, the TETRA development group has adopted several changes to its processes, to accommodate the Defect Prevention environment.

Internal dissemination outside of the TETRA development group, has yet to be done and will begin with the presentation of the Defect Prevention method to the SEPG - Software Engineering Process Group, the owner of MCIL's OSSP. This group will analyse the results of the PIE project, and update the OSSP accordingly. The SEPG will also be responsible for deploying the new process and training the other development groups. This will be done through a series of technical meetings with engineers and managers, dealing with Defect Prevention, the PIE and the updated OSSP.

Measured Results

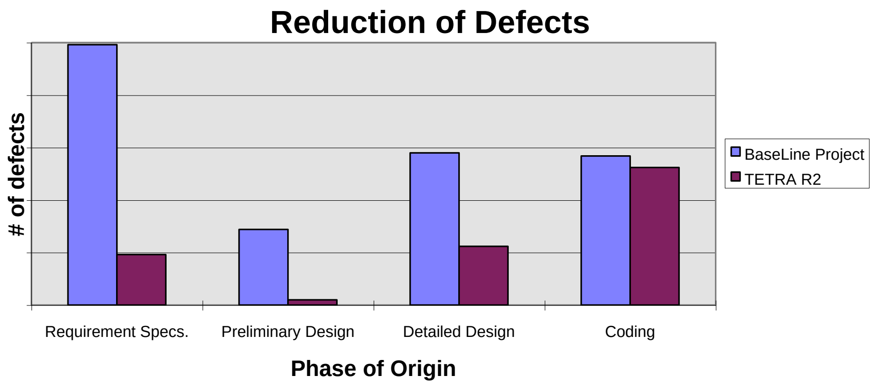
The overall number of defects in Tetra Release 2 has **decreased by 60%,** in comparison to the number of defects detected in TETRA Release 1 (the reference line project). In part, this can be attributed to the fact that Release 2 is a continuation project and not an initial project as Release 1, and that later releases usually have less defects due to more cohesive teams, greater familiarity with the application domain, experience, and fewer undefined issues.

Based on numbers from other MCIL projects, we estimate that half of the defect decrease (30%) can be attributed to the implementation of the PIE.

A breakdown of the defects, by Phase of Origin, shows the following results :

	TETRA Release 2	Past Projects	% Improvement
Phase of Origin			
Requirement Spec.	**20%**	**40.8%**	**80.6%**
Preliminary Design	**2.5%**	**11.8%**	**93%**
Detailed Design	**23%**	**23.9%**	**61.4%**
Coding	**54.5%**	**23.4%**	**8%**
	100%	**100%**	**60%**

The absolute reduction in defects, which relates to the % improvement shown in the above table, can be observed in the following chart :



The obvious observation is that a higher percentage of the defects “migrated” to later phases of the development process : from Requirement Specifications, Preliminary Design and Detailed Design, to Coding. In Tetra Release 1, 76.5% of the defects are in the Requirement and Design phases and only 23.4% are in Coding, while in Tetra Release 2, only 45.5% of defects are in Requirement and Design and 54.5% are in Coding. This implies that the defect prevention methods employed in the early phases of development were very effective.

The % Improvement column, shows the improvement within each development phase, with respect to the absolute number of defects. This is a different view of the improvement in the number of defects, partially attributable to the Improvement Actions.

Another comparison was made in respect to the Cause category. Following are the results :

Cause category	**TETRA Release 2**	**Past Projects**
Communication	**10%**	**11%**
Education	**13%**	**13%**
Oversight	**74%**	**74%**
Transcription	**3%**	**2%**

The obvious observation here is that the differences are **not significant**. The largest bulk of the defects are caused by human errors.

Lessons Learned

There are several key lessons learned from this PIE project :

1. Although Defect Prevention is considered an SEI/CMM Level-5 KPA, we found that a strong Level-3 organization, with a Defect Prevention infrastructure, can build an effective Defect Prevention Process, and obtain excellent results.

2. The primary cause of defects as classified by the development team is oversight, or human error (almost **75%**). Our experience shows that the term “oversight” is too broad and should be broken down somewhat, probably based on the Beizer classifications of those defects which were categorised as “oversight”.

3. The Timing of the Phase Kickoff meetings is critical. A Phase Kickoff should be planned early and performed as close as possible to the beginning of the phase.

4. In order for the Defect Prevention process to be effective, the software teams need in-depth training and initial support in using the taxonomy and performing the root cause analysis.

5. A tool to input the classification of defects, according to the Beizer Taxonomy is essential. An automatic tool is needed to analyze the defects and to get statistical results. The current vehicle we have for input of cause analysis and defect classification is deficient. A better interface is needed, as well as a mechanism for adding new categories to the Beizer Taxonomy. Standardized statistical analysis reports are needed for use by all projects for ongoing Defect Prevention and process improvement.

References

[1] R.G. Mays, C.L. Jones, G.J. Holloway, D.P. Studinski, “Experiences with Defect Prevention”, IBM Systems Journal, Vol 29, No. 1, 1990

[2] Watts S. Humphrey, “Managing the Software Process”, Chapter 17 - Defect Prevention, ISBN-0-201-18095-2

[3] Beizer Boris., “Software Testing Techniques”, Second edition, 1990, ISBN-0-442-20672-0

Appendix A - Authors’ CV

Paul Rogoway
Director of Software Quality Standards, Motorola

Motorola Experience (16 Years)
Coordinates all Motorola activities relating to software quality standards, including participation in national and international software standards working groups; identification, development and promotion of internal Motorola standards; and dissemination of information to Motorolans concerning risks and opportunities related to software standards.

Serves on Corporate Software Engineering Technology Steering Committee and on various international committees and working groups in areas such as software life cycle models, capability assessment, technology and tools, and process improvement. Israel's delegate to several international software standards organizations, including those responsible for SPICE and ISO 9000-3. Chairperson of the Israel Software Process Improvement Network (I-SPIN).

Member of Motorola's Science Advisory Board Associates (SABA). Winner of Motorola Outstanding Impact Award for contribution to ISO 9000-3 revision project. Authorized Lead Assessor in the SEI Appraiser Program.

Previously established and managed two software development groups, headed a Motorola Senior Executive Program team which produced a methodology for software engineering technology planning, founded and managed "4S" to help Motorola and non-Motorola organizations accelerate their improvement in software development capability, served as process improvement coach/consultant to several Motorola software development organizations and to leading non-Motorola software development organizations in Israel, and was responsible for the first phase of the CASE* project to define an Integrated Project Support Environment for Motorola worldwide.

Other Experience :
Adjunct Professor of Computer Science, Bar-Ilan University, Israel.

Before joining Motorola, Prof. Rogoway held senior technical and management positions at major U.S. and Israel companies, including TRW, Informatics, IBM, Elbit and Tadiran.

Publications :
More than 40 papers, articles, and lectures on various software engineering topics

Abraham Peled
Software Process Improvement Manager, MCIL

Motorola experience (12 years) :
Currently, manager of Software Process Improvement activities in MCIL's Development Division, which recently achieved Level 3 in a SEI CMM assessment. The chair-person of two internal Software Process related committees : The Software Engineering Steering Committee and the Software Engineering Process Group (SEPG). Responsible for and coordinating the activities of the various Process Improvement Groups.

A member of the Software Quality Committee (SQC), a corporate level committee, dealing with software quality issues, with Software Quality Metrics, and with deployment of the Quality System Review (QSR) within Motorola facilities, worldwide.

Previously, established and managed for 9 years the Systems & Software Quality Assurance, MCIL's Box and System Testing group, who is responsible for testing and releasing all the Software/Firmware products of MCIL. Most of the tests performed, were "shifted" from manual testing to the automatic testing environment, developed internally over the years. Also, Manager of the PQE group - who is responsible for Hardware and Environmental testing of all MCIL's released products. Responsible for the planning, installation and maintenance of MCIL's conventional SmartZone System, and for the new Digital Tetra/Dimetra System.

Leeba Salzman
Software Quality Manager, MCIL

Motorola experience (12 years)
Currently Software Quality Manager in the Digital Radio Group, responsible for process adherence, new employee training, and process improvement deployment. Member of the Software Engineering Process Group (SEPG). Responsible for MCIL training on Software Peer Reviews, Problem Tracking and DDTs. Head of MCIL DDTs Empowerment Team.

Abraham Danon
Systems & Software Quality Assurance Manager, MCIL

Motorola experience (16 years)

Currently, manager of MCIL's Software Box/System testing group. Member of MCIL's Software Engineering Steering Committee and editor of MCIL's publication on Software Engineering : "Touch Of Quality".

During years with Motorola, carried on technical and managerial software tasks including responsibility for software process, tools and technology.

Appendix B - Company Profile

Motorola Communications Israel, Ltd. (MCIL) is an industrial company which develops and manufactures radio equipment and radio related products. MCIL's Development Division is one of Motorola's largest development group outside of the United States, and is involved in many main-line development projects of Motorola's Communication Enterprise (CE).

MCIL is a fully-owned subsidiary of Motorola Inc., the world-wide leader in wireless communication. Motorola develops and manufactures components, products and systems in the following domains : Semiconductors, Cellular Systems, Paging Systems, Two-way Radio Communication, Modems and Integrated Management Systems, Automotive Electronics, Government and Space Systems (Irridium), and Multimedia.

According to the latest "Fortune 500" list, Motorola is the 29th largest company in the U.S., with revenues of $30B in 1997, and 150,000 employees worldwide.

Practical Implementation of a CleanBase

Malgorzata Warne
Ericsson Radio Systems , Sweden

Product Description

Cello is a system platform with an ATM (Asynchronous Transfer Mode) switch used by the wide-band cellular telephone system using the WCDMA (Wide-band Code-Division Multiple Access) world-wide radio standard.
Cello includes a real-time multi processor system, miscellaneous ATM services (based on our own developed ATM switch) and operation & maintenance support as well.
Recently, Cello has been delivered, as part of the experimental system, to one of the biggest telecommunication company in the world, NTT in Japan. As soon as next year, Ericsson a fairly new commercial system with new base stations and new transmission solutions will offer NTT and other customers.

Starting Scenario

There is a unique situation: a new world-wide radio standard is born and a new product generation is under development. The market window has opened now and a major development organisation (over 1,000 people involved) has started.
Our department (over 130 people involved) is developing a communication switch platform, Cello. We are a “Subcontractor” for a development of mobile radio network.

There is an outstanding opportunity for the department: Cello is a brand new, high-tech product.
At the same time, this is a huge responsibility for the project management: the delivery time is not negotiable and the product quality should minimise/exclude the need for product support in the future.
There we have the CleanBase Process.

Plans and Expected Outcome

All employees in the department are very familiar and pleased with incremental development and teamwork. No one wants to “go back” but, on other hand, no one has

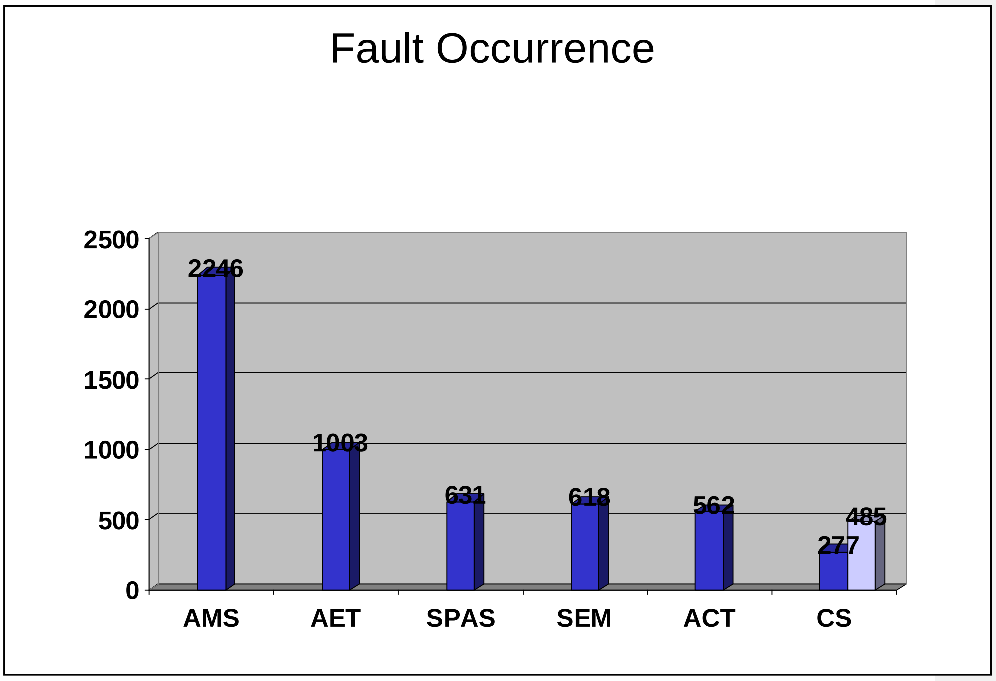
time with improvements such as more frequent or improved code reviews.
I discussed the situation with the system integrators who have been subjected to our delivery quality. We decided to do a very basic measurement afterwards: to count error occurrence, in other words, a number of code lines divided with the reported code faults per subsystem. I.e. in the ACT subsystem SW, statistically every 562 lines there was an error. The AMS subsystem had the least number of the reported faults and the CS had the highest number of the reported faults.
The graph layout of the collected metric values was perhaps not so professional, but still showed the arrangement of metrics (see next page).
Our goal was to get a motivation factor for improvement for all project members. Then we presented the metrics results at the project meeting and found out that the presented metrics data was really appreciated.

The metrics indicated a very clear tendency: the reviews contribute to higher product quality by preventing defects.
During the project, the members of the subproject, which have had the highest fault occurrence in their SW, have had no time for reviews. In fact they have had it very hectic, but unfortunately it was their consensus that they have had no time for reviews. In reality, just this particular subproject was delayed several times with their incremental deliveries (only a few days, but still) and their delivery quality needed some correction releases of the product.
It was a great pleasure to see and give the evidence to others that the theory and practise are in accordance.

At the same time I was picking up the metrics, I interviewed subsystem representatives to get a broader picture of code reviews usage. I asked them five simple questions:

- Have your teams done code reviews?
- Have your teams had some code review template?
- Have the reviews been incremental during one assignment?
- How have your teams experienced the reviews: as a quality increases, as a help to sharing knowledge or was it a waste of time?
- Have your teams some proposal about improving the code reviews?

The common opinion in all answers was very positive to code reviews and their outcome.



Actions

Now our department is involved in the upstart of a new project with a goal to develop a commercial product after we completed our experimental system.
All subproject managers discussed the metrics results (from the earlier project) at their project meetings with a goal to carry out the review improvements.
Simultaneously, we have planned a few metrics to follow the progress in the code review improvement as a goal and objectives for the project quality plan.

Measured Results and Lessons Learned

The measured results are:

- The code reviews were very useful in sharing knowledge between team members and contributing to the "next best" competence within the department.
- Initially, the reviews demand discipline and some time, but in the long run you get better delivery precision with a predictable quality.

The lessons learned are:

- The first step for the successful SW improvement is to implement an improvement's need in the consciousness of the project members and than they will support you with willingness for future improvements.
- Already the first simple measurement gives an outcome that is a base for future improvements.

Appendix:

Ericsson is the leading provider in the new telecom world, with communications solutions that combine telecom and datacom technologies with the freedom of mobility for the user. With more than 100,000 employees in 140 countries, Ericsson simplifies communications for its customers – network operators, service providers, enterprises and consumers – the world over.

Reuse of software development experience at Telenor Telecom Software

Magne Jørgensen,
University of Oslo, magne.jorgensen@ifi.uio.no

Dag Sjøberg
University of Oslo, Dag.Sjoberg@ifi.uio.no

Reidar Conradi
The Norwegian University of Science and Technology, Reidar.Conradi@idi.ntnu.no

Abstract

In this paper we describe how Telenor Telecom Software (TTS) developed and implemented processes, roles and tools to achieve reuse of estimation and risk management experience, i.e. organizational learning. The results from the case study include:

- the development and introduction of an experience database integrated with the software development process – offering relevant experience "just in time"
- examples of types of experience useful for software developers
- recommendations on how to collect, package and distribute experience
- experience on roles and process to support reuse of software development experience

Key words: Experience database, software improvement, organizational

learning

1 Introduction

The reported case study on reuse of software development experience was carried out in 1997-1998, supported by the national research project SPIQ (Software Process Improvement for better Quality). The case study was, among others, motivated by the following challenges:

1) How can software development experience be efficiently shared between different development teams?
2) What types of experience are worth reusing?
3) What is the role of reuse of "local" (context-dependent) experience compared with more "global" (best practice) experience?

Our approach and results to help meeting these challenges, we believe, can be useful for other organizations facing similar challenges.

The remainder of the paper is organized as follows. First we describe the research project SPIQ, then organization studied. Section 2 describes and argues for the approach chosen. Section 3 describes the results. Section 4 describes related work. Section 5 concludes, summarizes and suggests further work.

Software Process Improvement for better Quality (SPIQ)

In April 1997, following a pre-project in 1996, the software process improvement project SPIQ started. The program is sponsored by the Research Council of Norway (NFR) for at least three years. Its main goal is to:

”increase the competitiveness and profitability of Norwegian IT-industry through systematic and continues process improvement ....”

The SPIQ project is based on the software process improvement principles of “Total Quality Management”, see for example [10], and the “Quality Improvement Paradigm”, see for example [2]. An important aspect of SPIQ is that it provides a means for the academia and the software industry to meet and discuss software improvement experiences and research results. The work described in this paper has benefited from SPIQ in at least three ways:

1) The experience database design and results were discussed at the SPIQ meetings.
2) SPIQ has provided valuable research support.
3) SPIQ has financed parts of the Telenor Telecom Software’s (TTS's) internal work on “reuse of experience”.

The organization

TTS is split into five geographical locations and has more than 400 employees, most of them software developers. In other words, reuse of software development experience is an important but not trivial task. In 1995-1996 the company went through a “Business Process Reengineering”, see [14], resulting in a well documented, standardized software development processes. The process descriptions and documents are available to all employees through the Intranet using an Internet browser.

The software development process used by the developers is called “solution delivery” and is based on incremental delivery of software functionality in so called “time-boxes”. Each “time-box” lasts 3-6 month, which provides good conditions for experience reuse, at least compared with organizations with a waterfall development model leading to projects with cycles of 1-2 years. The organization includes several support teams (development tool support team, measurement and estimation support team, test support team, quality team, etc.) for the development and maintenance processes. These teams turned out to be very important in the implementation of the process changes and collecting experience. A recent, informal, in-house assessment (carried out by one of the authors of this paper) of the company, in accordance with the CMM framework, gave maturity levels on different key process areas between 2 and 4, i.e. TTS is a reasonable mature software development organization.

The company’s software development process prescribes several steps motivated by the need for reuse of development experience: Each project should 1) be measured according to a measurement model and 2) deliver an experience reports when completed. The "Measurement and Estimation Team” was allocated to carry out the measurement and the "Quality Team" was the receiver of the experience reports. We found that the project measurement and the experience reporting were to some extent carried out. However, there was not much systematic use of the information to improve the process. This observation was a major motive for our focus on reuse of experience in TTS.

2 The approach

Our approach can be characterized as action science [1], which is a typical research method when studying industrial software development. Action science has both advantages and disadvantages. Advantages are, for example, that action science may be the most efficient way to get:

- In-depth knowledge about software development organizations. This belief is among others supported by the learning model of [11], which focuses on the role of collecting concrete and context-dependent experience to support the learning process. According to this learning model only the lower levels of knowledge is context-independent and rule-based. In order to achieve higher levels of knowledge (being an expert) lots of context-dependent experience (local experience) have to be collected. *Our observations support this learning model. For example, while inexperienced project leaders asked for rule based methods regarding risk management, more experienced project leaders were more interested in how other projects had carried out their risk management activities.*

- Representative and realistic information on how terms and models important for meaningful reuse of experience are used. *For example, when we cooperated with the projected leaders on estimation of effort, we found a variety of interpretations of the term "effort estimate". This variety clearly reduced the potential for reuse of the effort estimation experience and data. Three major types of interprations were found: Estimated effort means a) "most likely effort", b) "the effort with the probability of 50% not to exceed" (median) or c) "the most likely effort + a (project dependent) risk buffer".*

Disadvantages of action science are, on the other hand, that:

- Action science studies are not carried out as strict experiments with control of the variables. Thus, a formal cause-effect relationship between the actions and the results cannot be established. In particular, the mixing of the participation and observer role makes objective analyses difficult. In addition, it is unlikely that anyone will (be able to) repeat the study to validate our observations.

- There is no available observational language or theory to remove subjectivity and bias in the description of the observations. See for example the discussion of how the expectations impact the observational language in [13] — i.e. there is a danger of "theory loaded observations".

It is important to be aware of these disadvantages, but it should not stop anyone from carrying out studies like ours. Currently, action science (or similar methods) seems to be the only practical way of achieving in-depth "real-world" results about software improvement. We believe, however, that more quantitative and experimental research on software processes should be the long-term goal of the software improvement research, leading to more general and objective knowledge. A more general discussion and comparison of research methods, particularly the role of case studies, can be found in [12].

Stimulated by the work at NASA-Software Engineering Laboratory on Experience Factory, see for example [4] and the opportunities we had at TTS, we started a search for "pilots" where reuse of experience would improve the development process. Based on an informal analysis of the availability of information, availability of resources, time, probability of success, estimated cost and benefit we decided to focus on the following two topics within the software development process:

- estimation of software development effort
- risk management

A brief analysis gave that in order to support reuse of estimation and risk management experience, there was a need for:

- an experience reuse process, including new or modified role descriptions
- a supporting tool (the experience database)

3 The results

- allocated experience reuse resources, both for implementing the experience reuse processes and for administrating the experience database

This section describes the work and some of the results achieved in the period Spring 1997 - Spring 1998. The organization continues to focus on experience reuse, i.e. the results and products are to some extent preliminary.

Manifestation of experience

During the requirement analysis we soon discovered that the manifestation of experience can and should take many forms to be useful to the developers, such as:

- quantitative and qualitative information that can be stored in traditional databases.
- general tools implementing or based on "best practice" within the organization
- rule based systems (expert systems) reflecting expert experience and knowledge
- pointers to people with useful experience (this may be the only way of "representing" experience that cannot be articulated, i.e. tacit knowledge)
- process descriptions at different levels and with different degrees of context dependence

In addition, it was considered important that the experience database (the tool enabling the access to the stored experience) was available to all the developers at a low cost, integrated with the quality system, easy to use and easy to maintain.

Technical platform

The technical platform chosen to meet these requirements was based on:

- The organization's own Intranet. This made the experience database available to all the developers and well integrated with the organization's quality system.
- A user interface based on a web-browser with links to experience of different types. This removed the need for local installation.
- An "experience database" based on tables of data, spreadsheets, documents and rules implemented in executable programs, i.e. no traditional database.

Further, we decided to integrate the experience reuse support with the organization's process descriptions, i.e. from the relevant steps in the process descriptions we had links to useful information and tools in the experience database. The idea was to offer useful experience "just in time".

Reuse of effort estimation experience

The effort estimation experience we offered was of the following types (linked to the relevant process steps):

A) Determine the appropriate estimation model and process.
An “expert system” recommending one or more estimation models was developed based on the collection and analysis of the experience of the organization’s estimation experts. Following an analysis of whether formalized effort estimation is recommended or not, the expert system asks the user to answers nine questions. A simplified description of the questions and some implications of different answers are indicated in Table 1 and 2. The estimation models are briefly described below. This expert system uses, in addition to the answers from the users, empirical data from TTS on the accuracy of the different estimation models, see Table 3, and the quality of the relevant historical data, i.e. a high degree of organizational dependent experience.

Table 1

Questions (Yes or No-answers)
Q1) Will there be a high degree of infrastructure development and/or complex algorithms?
Q2) Is the project context significantly different from previous TTS-projects?
Q3) Are most of the requirements described?
Q4) Is a data model available or can easily be developed?
Q5) Does the delivery consists of many small, not logically connected changes/modules?
Q6) Will the effort to complete the project probably be more than six months?
Q7) Is the project willing to spend 1-2 man-days of effort on estimation for small project (less than 12 man-months) and 2-4 man-days for larger projects?
Q8) Will developers with experience from similar projects be available when estimating the effort?
Q9) Will there be more than five deliveries similar to this one?

Table 2

Estimation model	**Recommendation rule** *(Qi has the value TRUE when the user answers YES on question Qi, and the value FALSE given the answer NO. If Ri is assigned the value TRUE, then model Ei is recommended.*
E1) FPA	R1 = not Q1 **AND** not Q2 **AND** Q3 **AND** not Q5 **AND** Q6 **AND** Q7
E2) ROPD	R2 = not Q2 **AND** Q8
E3) FPA simplified	R3 = not Q1 **AND** not Q2 **AND** Q4 **AND** not Q5 **AND** not Q7

E4) Tailor made estimation model	R4 = Q2 **AND** Q9
E5) No model recommended	R5 = not (R1 **OR** R2 **OR** R3 **OR** R4)

When more than one model is recommended we give the estimation model with the lowest index the highest recommendation and presents the other recommended models as alternatives.

Table 3

Estimation model	**TTS historical accuracy of model (average)**
Full MarkII Function Point Analysis	15% (mean magnitude of error)
Simplified Function Point Analysis	30% (mean magnitude of error)
ROPD	20% (mean magnitude of error)

B) Estimate effort
Depending on estimation model, different types of experience data are available.
Among others, the following estimation models and planning tools were supported by the experience database:

1) MarkII Function Point Analysis (MkII FPA), see [22]. We improved and extended an existing spreadsheet implementing the MkII FPA estimation model. This estimation model takes as main input the estimated size of the functionality to be developed in function points.

Earlier we had analyzed data from more than 30 software development projects regarding how different variables, such as use of CASE tool, had had an impact on the development productivity, see [15]. This study indicated that the choice of development environment explained most of the productivity variance.

(For example, an effort estimation model for Cobol and Powerbuilder-projects, based on log-linear regression on the collected data, including only the size of the task and the development tool "level" as independent variables gave a R^2 of 0,52.)

We provided the estimator with historical data on previous projects similar to the current project. Table 4 shows some of the historical information that the estimator could make use of. The productivity is measured as UFP/w-h, unadjusted function points per work hour. Notice that the estimator has to predict a productivity category for his project, i.e. expert knowledge is still required.

Table 4

Batch-development	**Low prod.**	**Medium prod.**	**High prod.**	**Turbo prod.**
Cobol – environment	0.05 UFP/w-h	0.10 UFP/w-h	0.20 UFP/w-h	0.30 FP/w-h
Powerbuilder – environment	0.15 UFP/w-h	0.25 UFP/w-h	0.50 UFP/w-h	0.70 UFP/w-h
On-line development	**Low prod.**	**Medium prod.**	**High prod.**	**Turbo prod.**
Cobol – environment	0.07 UFP/w-h	0.15 UFP/w-h	0.20 UFP/w-h	0.30 UFP/w-h
Powerbuilder – environment	0.20 UFP/w-h	0.35 UFP/w-h	0.70 UFP/w-h	1.00 UFP/w-h

2) A bottom up, task and risk based estimation model was developed. This estimation model was supported with experience in the form of lists of "tasks to remember" and suggestions on the effort distribution between the phases. Currently, there is ongoing work on how to improve the collection and reuse of historical data to support this bottom-up, task and risk based estimation model, see [20]. We labeled this model **ROPD** (the Norwegian acronym for Risk Based Division into Sub-tasks).
3) A risk analysis tool integrated in the estimation tools (or to be used separately) was developed. The risk analysis tool contains risk models, textual advise and guidelines based on previous experience. The content varies from a simple (but useful) checklist of tasks and risk factors to more sophisticated probability (beta-distribution) based risk models. Typically, the content was based on general frameworks and models, then adapted to the organization's needs according to expert knowledge and experience. This tool resulted in a probability based effort estimate and predictions such as "there is an 80% probability of not exceeding 3000 w-h of effort".
It turned out that this type of probability based predictions were essential to introduce the distinction between planned and estimated effort in the organization. Similar to the results in (Conolly and Dean 1997) we believe that probability based estimation had a positive impact on the realism in the effort estimates.
4)Finally, pointers to the human estimation experts were provided.

Reuse of risk management experience

Similar to the estimation support we linked our experience database to the risk management process. The experience database offers support through several tools to identify, analyze and manage software project risks. We interviewed several experienced project leaders in the organization to get the most relevant risk factors and the most relevant methods to reduce and

control the risks. In addition, data from quality revisions was used to tailor the risk management support.
Based on the collected information we developed:

- a "TTS best practice" risk management process (extensions to the existing development process)
- a tool to identify, assess and store risk factors, and suggestions on how to reduce or control the risks
- a tool to visualize the risk exposure over time

In many ways, what we did was to collect only a small fraction of the organization's knowledge about risk management. To become a learning organization the organization will need to continuously collect and distribute experience, i.e. new roles and a changed process is needed. Since systematic experience reuse in risk management has a short history in TTS, we found that we needed to start small in order to understand what sort of risk experience would be useful to collect.

Roles and process

The studies and results described earlier in this paper resulted in the identification of needs for new roles and an increased focus on the implementation of the development process.

Roles:

- An "**experience database administrator**" (a "gardener") responsible for the availability and usability of the experience to be reused. This role may be split into two roles dividing the responsibility into a technical administrator and a content administrator. We suggest that the "gardener" should be a part of the software process improvement team of the organization.
- Several "**process analysts**" responsible for analysis of information from each sub-process, such as the estimating process, the project management process or the testing process. The "process analysts" is responsible for collecting and analyzing relevant information from completed projects and to generalize, tailor and package the useful experience.
- A network of "**support teams**" teaching and guiding the project leaders and members how to properly reuse the experience within each sub-process/topic.
- A **process owner** for the experience reuse process.

Notice the distinction between role and person. In a small organization a small team or (at least in theory) one single person may fill all these roles. Based on our experience at TTS, a critical minimum central effort to enable substantial reuse of estimation and risk management experience seems to be 2-3 man-years to fill the roles above.

Process

When we started our study, the organization did collect project data and it was mandatory to write experience reports, i.e. the process description had elements of experience reuse. However, the collected information was not systematically used to improve the processes. In other words, the process (or even more, the implementation of the process) had not had enough focus on the use of the collected information. Looking at other case studies of software process improvement, see for example [8], this seems to be a typical problem leading to graveyards of data and unused documents. In our opinion, this is a situation even worse than the situation where no data is collected and no reports written, and there is probably no more efficient way of destroying the respect for a measurement and experience report.

We believe that the current process description of TTS is sufficient to enable experience reuse, given sufficient resource to fill the experience reuse roles described earlier. For a more general experience reuse process and organization, see [4].

Benefits

An underlying initial hypothesis on experience reuse is, of course, that it has a long term benefit higher than the costs. Currently, we are not in the situation to decide whether this is true or not. We cannot validate the hypothesis, partly because it is too early, and partly because it is difficult to isolate the impact of our work from the impact of other parallel process improvement initiatives. However, even without a formal impact study, we believe to see the following results of the experience reuse work:

- Improved estimation accuracy and more widespread use of the estimation models
- An increased focus on experience based risk management in the projects.
- An acceptance in the organization for the need to collect and share experience

In addition, we have made a number of interesting observations increasing the probability of sucessful reuse of experience in TTS, such as:

- Currently, the experience reports written by the projects were of little use to other projects. This may indicate that without a clear model on how the experience will be reused, there is a great danger of reporting and collecting useless information.
- The mere focus on reuse of experience had a positive impact on the "improvement culture" in the organization. It would have been very interesting to carry out controlled experiments on how different actions impacts the software improvement culture. An experimental design similar to the one described in "Goals and performance in computer programming" [23] may be appropriate.

4 Related Work

The Experience Factory or EF [4,5] is a framework for reuse of software life cycle experiences and products. EF relies on the Quality Improvement Paradigm [3] for continuous and goal-oriented process improvement, resembling the Shewhart/Deming Plan-Do-Check-Act cycle [9].

The EF framework prescribes an improvement organization inside a company, a kind of "extended quality department". This implies the "logical separation of project development (performed by the Project Organization) from the systematic learning and packaging of reusable experiences (performed by the Experience Factory)" [6].

The PERFECT EF framework extends this model by adding a third organizational component: the Sponsoring Organization, which uses the EF for strategic purposes [18].

Within the EF framework, the NASA-Software Engineering Laboratory with its 275 developers has collected information about 150 projects in the period 1976-1996. The purpose is to record the effects of various software technologies (methods, tools, programming languages, QA techniques, etc.). However, NASA represents a special kind of stable and resourceful organization. It is a challenge to apply the EF ideas outside of NASA, i.e. to downscale it to companies with typically 10-30 developers, and where the EF roles are partly being played by the developers themselves. More applications of the EF framework in other contexts are therefore needed, see e.g. [18]. Our case study is a contribution in that respect.

5 Conclusions

We believe to have contributed to the answers regarding the challenges we described in Section 1 through an in-depth example of how the questions/challenges were approached by TTS. TTS has introduced a standardized development process documented on the web and made the processes available for all the software developers through the organization's Intranet. In many ways, this opens new possibilities for software development organizations. We have found that software development experience efficiently can be linked to the process steps and made available to all the developers in a very flexible way. However, the main challenges regarding becoming a learning organization and reusing experience is not the technology. We found that a lot of "trial and error" and pragmatism is needed to find the useful experience and ways to formulate and spread this experience.

We found it useful to be very pragmatic regarding the manifestation of experience. For example, a very useful information in our experience database was the links to the experts having the required experience. Regarding the role of local (organization dependent) experience vs. best practice experience we found that the local experience made the best practice processes significantly more useful. In other words, optimal use of best practice processes seems to require collection and reuse of more local experience.

Achieving a learning organization is a formidable task. Senge claims that the following five disciplines are essential to creating learning organizations: *personal mastery, mental models, shared visions, team learning* and *systems thinking* [21]. An experience database like the one we have designed and implemented in TTS can serve as a basis for activities involved in all five disciplines. An experience database is also a useful means to agree on a common understanding of the current situation. "*An accurate, insightful view of current reality is as important as a clear vision*" [21].

Future work will address the major issue of how projects (contexts) should be characterized so that experiences collected in one project (context) are applicable to another project (context). How can we judge whether a project is sufficiently similar to (a subset of) the projects for which we have experience? The approaches described in [6] will be taken as a starting point.

Acknowledgments

The authors wish to thank the TTS employees Pål Woje, Geir Ove Espås, Majeed Hosseiney, Oddmar Aasebø and Tor Larsen for their enthusiasm and

References

contribution to the work described in this paper.

[1] C. Argyris et al, 1985, Action Science: Concepts, Methods and Skills for Research and Intervention. San Francisco: Joosey-Bass.

[2] V. R. Basili, 1985, Quantitative evaluation of software engineering methodology, *Proceedings of the First Pan Pacific Computer Conference*, Melbourne, Australia.

[3] V R. Basili and H. D. Rombach, 1988, The TAME Project: Towards improvement--oriented software environments, *IEEE Transactions on Software Engineering*, vol. SE-14, pp. 758–773.

[4] V. R. Basili et al. 1992, The software engineering laboratory — An operational software experience factory. In: *Proceeding of the 14th international conference in software engineering*, Melbourne, pp. 370-381.

[5] V. R. Basili, 1993, The experience Factory and its Relationship to Other Improvement Paradigms, pp. 68-83, in I. Sommerville and M. Paul (eds), *Proc. From ESEC'93, 4th European Software Engineering Conference*, Garmisch-Partenkirchen, Germany, September 1993, Springer-Verlag, Lecture Notes in Computer Science 717.

[6] V. Basili, L. Briand, and W. Thomas, 1994, Domain Analysis for the Reuse of Software Development Experiences, In *Proc. of the 19th Annual Software Engineering Workshop*, NASA/GSFC, Greenbelt, MD..

[7] T. Conolly and D. Dean, 1997, Decomposed versus holistic estimates of effort required for software writing tasks, *Management Science*, vol. 43, no 7., 1029-1045.

[8] M. A. Cusumano and R. W. Selby, 1996, *Microsoft Secrets*, Harper Collins Business, ISBN 0006387780.

[9] W. E. Deming, 1982, *Quality, productivity, and competitive position*. Massachusetts Institute of Technology Center for Advanced Engineering Study, Cambridge, Mass.

[10] W. E. Deming, 1986, *Out of the crisis*. MIT Center for Advanced Engineering Study, MIT Press, Cambridge, MA.

[11] H. Dreyfus and S. Dreyfus. 1986, *Mind over machine: The power of human intuition and expertise in the era of the computer*, Free Press, New York.

[12] B. Flyvebjerg, 1991, *Rationalitet og magt, det konkretes videnskap (bind I)*, Akademisk Forlag.

[13] N. Goodman, 1951, *The Structure of Appearance*, Cambridge, Mass.

[14] M. Hammer, 1996, *Beyond reengineering*, Harper Collins, New York.

[15] M. Jørgensen, 1995, Empirical evaluation of CASE Tool efficiency. *Proc. Sixth Int. Conf. on applications of Software Measurement*, Orlando, 207-230.

[16] G. Morgan, 1993, *Imagination — the art of creative management*, SAGE Publications, London, 1993, ISBN 0-8039-5299-6.

[17] M. Paulk et al, 1995, *The Capability Maturity Model, Guidelines for improving the software process*. Software Engineering Institute, ISBN 0-201-54664-7.

[18] PERFECT Consortium, 1996, *PIA Experience Factory, The PEF Model*, ESPRIT Project 9090, D-BL-PEF-2-PERFECT9090.

[19] E. H. Schein, 1987, *Process consultation (volume II),* Addison-Wesley, ISBN 0 201 06744 7.

[20] T. Schrader, 1998, *A bottom-up project cost estimation method using historic data and a standardized work breakdown structure*, Project Report, The Norwegian University of Science and Thechnology.

[21] P. M. Senge, 1995, *The Fifth Discipline: The Art and Practice of the Learning Organization*, Currency/Doubleday.

[22] C. R. Symons, 1993. *Software sizing and estimation, MkII FPA*, New York: John Wiley and Sons.

[23] G. Weinberg and E. Shulman, 1974, Goals and performace in computer programming. *Human Factors*, vol. 16.

Session 11

Personal Software Process implementation In a production environment

Stavros K Menegos, MSc

Computer Logic SA, Senior S/W Architect

Dr. Charalampos Avratoglou

Computer Logic SA, S/W R&D Director

Chapter 1: Introduction

Findings of the ESSI Process Improvement Experiment (PERSPI) will be presented, in the context of the overall effort of Computer Logic towards Software Process Improvement.

PERSPI is an experiment that aims at improving the way individuals perform their day-to-day activities in the context of software development. More specifically, PERSPI is an attempt to introduce the PSP methodology in an industrial environment, putting emphasis on its gradual employment in a real-life project, rather than on an introduction through formal and long-term training.

The presentation gives an overview of the environment on which the experiment is applied, i.e. Company’s business area and strategy, the objectives of the experiment and the baseline process characteristics. Next the PIE’s structure is given and the findings and lessons-learned are presented. Finally future actions and areas of investigation are discussed.

Chapter 2: Project Description

Business & Products

Computer Logic S.A.'s main function is to produce, market, distribute and support business application software products. The produced software products are:

- capable to adhere to a wide spectrum of needs and cultural environments,
- easily maintainable, upgradable and configurable (from the user's point of view),
- able to handle business critical applications.

Furthermore, the company's software development department

- undertakes custom software projects of variable size and complexity,
- studies and provides solutions and services to specific customer IS-related requirements.

During the last three years the company is dealing explicitly with the improvement of its development process, focusing initially on the software-engineering field, and then, on the organisational - managerial infrastructure of its software development process.

The above efforts where formed to support the business strategy for a development process based on State-of-the art technology, Software Reuse and Process Repeatability.

The PERSPI PIE has been designed to complement the above efforts, addressing the finest instrument of the software development process, the individual developer.

The starting scenario

The current mainstream development process is code-named the OMEGA process. The figure SME.1 presents a schema of this structure.

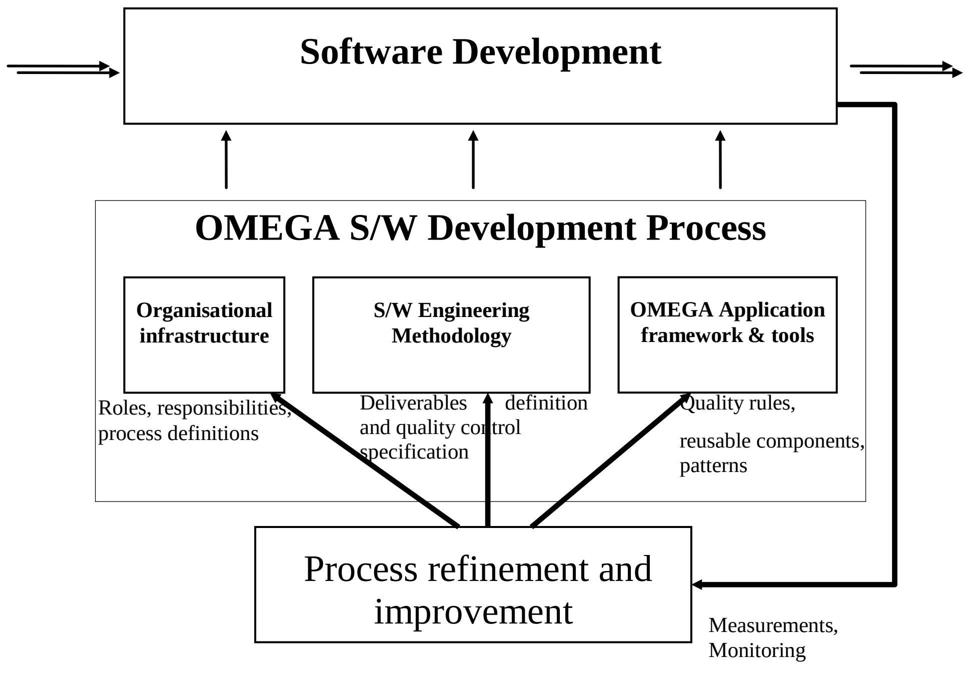

Software Development
OMEGA S/W Development Process
Organisational infrastructure
S/W Engineering Methodology
OMEGA Application framework & tools
Roles, responsibilities, process definitions
Deliverables definition and quality control specification
Quality rules,
reusable components, patterns
Process refinement and improvement
Measurements, Monitoring

Fig SME.1: Omega Process

In the software-engineering field a comprehensive OMT-based object-oriented methodology is currently used. A set of tools are used including an upper CASE tool, requirements-features-development items tracking database, a version control tool and an effort tracking and measurement tool.

The language is C++ and visual programming tools are used for the user-interface parts. Productivity is greatly enhanced by an in-house developed repository-based, OO-aware, dedicated application framework (the OMEGA framework) which encapsulates the generic application architecture and provides a set of reusable components. The framework also encapsulates a host of standardised quality rules that apply to the user and to the application developer interface.

The Organisational and Process infrastructure has been reorganised according to the CMM model and has been assessed as a level-2 with traces of level-3 process. Additionally the company is ISO-9001 certified for the development, maintenance and support of S/W.

The above top-down approach to process improvement, while promising and essential, seems to have low penetration to the micro-processes of individual developers, while it presents a significant cost for managerial overhead. On the other hand, the size of the development department and the traditional reliance on individual developers for quality and efficiency justify the need of improving the individual developer’s process, and this idea has immediately gained management commitment.

PERSPI Objectives

The main objective of PERSPI was to facilitate Computer Logic’s process improvement by applying software process best practices at the individual level.

It was anticipated that the successful introduction of Personal Software Process principles would improve both product quality, development efficiency and productivity by introducing best practices at the finest level of the development process, the individual developer.

It was also expected that this approach would directly involve developers in process improvement and would facilitate the institutionalisation of several improvement efforts done in the past in the software engineering, the quality assurance and the process management fields.

The following were expected at the end of the PERSPI project:

- Developers to be better able to define, measure and track their work.
- Developers to have a defined personal process structure and measurable criteria for evaluating and learning from their own and others experiences
- Developers to be better able to select those methods and practices that best suit their particular skills and abilities.
- Developers to be more effective and reliable members of their development teams and projects through a customised set of orderly, consistent, and high-quality personal practices.
- The results and experiences were expected to be fed back to improve the overall development process in order for it to support and promote individual best practices

- Based on the experiences and the measured results of PERSPI, a curriculum and a mentoring plan was planned to be prepared to apply personal best practice mentality through-out in the development department.

From the business point of view, the driving target was to improve product quality, predictability and development efficiency by enhancing individual developer process skills while reducing the managerial overhead. This improvement approach, at the individual level, is considered most compatible with the company's size and tradition.

The reference points and desired outcome determining the success of the experiment for the experiment have been set:

- to reduce by at least 25% the individual developers error rates,
- to increase the predictability for individual work products by reducing the slippage to +/-20%,
- to decrease time spend for a work-product by at least 10%
- to increase job satisfaction, personal commitment and process improvement awareness for the individual developers

The criterion for adoption of PSP at large, was set to be able to demonstrate that the investment per developer could be covered by the productivity gains with-in an approximately six-months period.

Chapter 3: PERSPI Experiment – Case study

The project of introducing the PSP in the Omega S/W Development process consisted of four phases. In the first phase a pilot team was assembled and acquired training to the PSP. In the second phase, the PSP process was evaluated in the context of the Computer Logic's needs and targets and in the context of the S/W development Framework and infrastructure. Upon the modification and customisation of the PSP the new process (modified PSP) was applied and monitored in the S/W development process of certain core projects. In the third phase the results of the PSP were assessed and evaluated according to the improvements in the quality of the delivered S/W, the productivity and the predictability. In the fourth phase the PSP was introduced into the process of *MIS Reports* of the Omega development life cycle. The process of the development of MIS reports on top of the Omega applications has many common characteristics with the development of S/W code and is of strategic importance for the success of the applications and for the achievement of high degree of customer satisfaction. Thus, an improvement in the quality and productivity of the individuals involved in this process was crucial.

Phase 1: PSP Training

In order to be able to apply PSP in our company development process we had to acquire training on the PSP. For this purpose, we selected the advance academic course for PSP given by the Carnegie Mellon University, Distance Education Program. The course is specifically designed for professionals and is given over the Internet. The duration of the course was eight weeks, which in our case was expanded to ten weeks. The course consisted of eight lectures and seven lab projects that had to be submitted every week. One significant advantage we had with this program was that we were allowed to form our class of students that attended this course (Computer Logic's class). Every week there was a chat session (over the Internet) between the class members and the Professor and his assistant who were assigned to the class. The purpose of this session was to discuss any problems with the PSP, address questions to the professor and in general to exchange ideas and considerations on applying PSP in a production environment.

The next step was to select the developers that would form the PSP class and participate in this distance learning course. We selected five Senior S/W Engineers that were involved in the development of core S/W Omega projects. In this team, the Omega Framework S/W Architect was also included. This role is responsible for the Analysis, Design and Implementation of the key features of the Omega library Framework on top of which all the Omega applications are implemented. All of the engineers had a very good experience in C++ programming language and in the development of high-quality S/W products under strict deadlines. They were also heavily involved in the previous Process Improvement efforts, mainly the CMM Process Improvement Project and the ISO 9000-1 certification. That means, that the necessity for the process improvement in the finest level of the developer was realised and very well understood by the whole team.

The participation in the PSP class required a significant amount of time and effort for the successful completion of the course. The estimated time was twenty hours per week per developer but the actual effort averaged to twenty-five hours. However, the training took place during a very demanding period, which did not allow for proper buffering of the developers from their projects. Thus commitment and self-sacrifice was necessary for the success of the training.

One of the advantages of having the education program in parallel to the work was that even from the first lectures on the PSP the developers applied certain aspects and guidelines in their everyday S/W development practice. This was a promising issue for the success of introduction of the PSP in our production environment.

A very important criterion for the construction of the pilot PSP team was the ability of the team members to train the whole Omega S/W development community to the PSP in the context of the existing practices and needs. For this purpose, we assembled a second team of five S/W Engineers who were recently introduced in the Omega S/W development process. Two members of the first pilot team trained this second team on the PSP. The education program for the second team started four weeks after the commencement of the PSP course. The reasons for which we established this second team were:

- Assess the ability and the difficulties of training the developers of the Omega S/W development process by the members of the pilot team.
- Identify the problems and the intricacies of introducing PSP in the every day practice of Junior S/W Engineers. Understanding the way junior or recently introduced developers think and tackle the implementation of S/W products (academic vs. production environment).
- Smoothly introduce the junior developer members of the Omega development process into the required discipline in both the personal and team level. This is an important issue since developers usually face difficulties in conforming to guide lines, discipline and company wise standards.
- Construct a broader team of developers that can share ideas and discuss self-improvement issues in the context of the Omega framework and PSP.

The remote training of the team had very good results. The participants managed to complete all the study and the course-works in due time in parallel to their full scheduled product work (Appendix B contains certain indicative results from the PSP training course). This is a critical point because full commitment and understanding of the responsibility the developers undertake is required for the completion of the training. The chat sessions proved very helpful because they gave the opportunity to share ideas between the academic and the professional perspectives of PSP. One problem we did face with the training material is that the programming exercises and the programming environment used in the course did not correspond to the tools and programming environments that are in use in modern production environments. Even though the exercises have the purpose to demonstrate PSP principles they could have been more complex and realistic. These thoughts and suggestions have been shared with the PSP course staff and they plan to modify the structure of the lab work projects.

As far as it concerns the training of the second team, this was a more difficult task to complete. The main problem was that the commitment of the participants to this effort was not as high as necessary. The result was that only two of the five developers actually completed the course. The second problem was that the instructors (two developers from the pilot team) did not have the available time to thoroughly examine the submitted lab work. However, as explained in the previous paragraphs this training effort gave us better insight on the possible implications we would meet when introducing PSP in junior members of the Omega development process. Actually we reached to the conclusion that the PSP training should be combined with the education – training of the new comers in Omega processes and Infrastructure. Another interesting conclusion is that having the PSP training performed by external institutes through distant learning is a better motivation for the participants for they are individually awarded.

Phase 2: PSP Evaluation – Modification

The next phase in the PERSPI experiment was to evaluate the PSP in the context of the Omega Development Framework and Process. For this purpose a committee was assembled that consisted of the five members of the pilot team, one member from the second team and the R&D Director who is responsible for the Omega Development process. The task of the committee was to thoroughly examine the pros and cons of the PSP and how the PSP will be modified and introduced into two kernel projects of the Computer Logic.

The main characteristic of the Omega S/W development process is the ability to quickly address the diverse needs of the enterprises and to develop competitive products for a wide range of business domains. The development process is assisted and supported to large extent by our custom developed library framework on top of which applications are developed.

The PSP as a means for improving predictability, productivity and quality is very promising. However, introducing and integrating the PSP as described in theory our development process and framework has certain advantages and disadvantages. The cons and pros are summarised as following:

Cons

- The volume of the data that the developer has to gather and maintain during the development cycle is very large. This does not only increase the overhead of PSP but also increases the disruption of the developer from his development tasks. The everyday process of writing code differs significantly from the well-established and defined examples that are given during the training courses or encountered in the introductory projects. In the latter, PSP tools are easy to apply and use, while in the former developers seldom have the time to use these tools when dealing with complex problems under strict deadlines. This issue was thoroughly considered and discussed in the chat sessions with the course's academic staff and it seems that it is the first and main issue for arguing for using PSP.
- Even the use of tools such as spreadsheets, text documents and database tools does not facilitate the extraction of the raw data PSP requires, especially data that have to do with Lines Of Code (LOC).
- The PSP assumes the existence of a well-defined and complete analysis and design prior to the implementation. While this is a sound prerequisite, it is not usually satisfied in a production environment such as Computer Logic's that has to address as fast as possible to a wide variety of customer needs in a very competitive market. The model implied in the context of PSP is more applicable in S/W Projects that the cost of development is very high justifying an extended cycle in the Analysis and Design phases. Typically not the case for Computer Logic.
- The PSP requires the existence of tools that will facilitate and automate the processing of the raw data as well as the historical data. These tools should be integrated to the existing tools and furthermore increase the cost of the

development. Another alternative that is discussed in later section is to build our own tools that address the modifications of the PSP methodology and focus on the issues we are interested in.

- Another point in the PSP that was a major issue for discussion is the concept of LOC and Object LOC. The most difficult part in applying PSP was the gathering of data that had to do with LOC and mainly the Base, Modified, Added and Deleted lines of code. This type of data is difficult and time-consuming to gather even with the assistance of automated Configuration Management tools that are used in the Omega Development process. Furthermore, we have reached the conclusion that in our case the size estimation in terms of LOC is not applicable. Even the Object LOC that is a higher level aggregation of LOC that defines the average Lines of Code per type of object, is not applicable. We have concluded that the categorisation of objects according to their difficulty and complexity is a better candidate for size estimation and is easier to count during or after the development cycle.
- A problem that is related to LOC issue is the code handled (inserted, deleted and modified) by the Source Code wizards that are in use in the Omega development Framework. These wizards handle the start-up code for the projects and for the various types of objects that are implemented. Furthermore, they do facilitate the insertion of code for typical cases e.g. handling events, handling User Interface elements, etc. This makes more difficult the counting and tracking of LOCs added, deleted or modified in the on going projects.
- Regarding the Reuse, PSP focuses in code reuse and mainly in LOCs implemented for reuse and objects reuse. However, in our Omega development process reusability is handled at the analysis and design phases. That means that the objects that exhibit reusable behaviour are explicitly designed for reuse either in the same project or in other projects. Thus the overhead and the gains of reusability are already taken into account in the early stages of development.

Pros

- The concept of Design and Code Reviews along with the *checklists* was considered as very significant and helpful for achieving high-quality S/W with minimised testing effort. The benefits of formally introducing Design and Code reviews and checklists are manifold. First of all, the lists could be easily combined with the Omega Framework Guidelines and the Computer Logic’s standards and code policy statements reaching thus a high degree of conformance and uniformity. An important characteristic is that the checklists always, due to their continuous refinement, reflect issues for the most common error patterns, the best practices and quality assurance aspects that assess the completeness and correctness of the designs and the delivered code. Furthermore, reviews can be easily performed on a regular basis, they have low overhead and they exhibit a high degree of ROI.
- In the context of the Omega S/W development process the compilation errors were not related to the quality and productivity. On the contrary, PSP considers compilation errors as an important factor that should be taken into account. The pilot team was totally convinced that this is true. Thus trying to eliminate or reduce the compilation errors at the coding stage would lead to higher quality products with less development effort in less time. It has been justified and proved in practice that writing code that has almost zero compilation errors implies mature and complete implementation. Furthermore, the time spent in

compilation was reduced by 20% depending on the size of the project – even now-days with the very fast compilers and fast machines, compilation is still a time consuming stage for product builds.

- Concerning Time-logging PSP addresses a very important factor that the Omega Development process had not considered in its Activity-Log (O Work In Progress System – WIPS) system. This factor is the number of interrupts along with the time spent on interrupts. Most of the developers of the Omega development process are involved in more than one project and impersonate more that one role. Thus interrupts are quite often and certainly have an implicit contribution to the number of defects.
- The discrimination and categorisation of defects as proposed by PSP is very promising and helpful. In general, PSP tackles very well the issue of tracking down defects in all the stages of the development (Defect Log List). The existence of historical data on defects is very crucial for the compilation and production of accurate and up-to-date Design and Code Review checklists. Furthermore, the Omega library framework would consider the most frequent defects and put an effort to incorporate work around solutions or even to eliminate certain type of errors. At the individual level, the Omega developer could study his defect patterns and categories of errors and eventually be more careful when designing or coding (self-imposed checklists).
- The categorisation of objects and the existence of historical data per type of object are very promising aspects of PSP. This classification enables the tracking of size estimation data, based on the complexity and the type of objects rather than relying on the Object LOC. The Omega development library framework, as described in previous sections of this paper, provides a set of reusable classes and constructs that facilitate the implementation of new objects for the Business Domain and the User Interface of the applications. The available objects and constructs are already classified in terms of estimated effort and complexity by the Omega. This classification could be easily transferred in the PSP context in order to improve the predictability of time and size.
- Unit Testing process and the Test results are handled in PSP in a very well defined and efficient manner. The Test templates and forms that introduced in PSP are similar to the Test scenarios being used in the Omega Development process. The vast majority of these test cases is derived from the Requirements Lists and the Use Cases description documents. However, PSP's Test templates are more thorough and formal and are dealing mainly with the validation rather than the verification of certain project's issues or behaviour.
- One implicit advantage of PSP is that explicitly introduces and requires a discipline at the S/W Engineering level that is difficult to enforce otherwise. The developers by nature tend to resist in rules and enforced guidelines. Furthermore, it is not company's will to have all the developers working and thinking in a predefined and identical way. On the contrary Computer Logic is relying on the individual's skills, capabilities and ideas that have put in the top S/W companies of Greece. The PSP gives individual developers the freedom to work as they used to while complying to certain guidelines and procedures that come from their own past data and by the expertise of their colleagues.

Having discussed and considered the aforementioned issues concerning PSP the next step was to modify the PSP in order to be easily applied in two kernel projects of the

Omega development process. The pilot projects that have been selected for the application of PSP were:

- *Omega Application Framework*.

  This project was selected because the framework is the foundation of all the Omega applications. The on-time delivery of high-quality Omega versions is crucial for the success of all the projects. Furthermore, all the guidelines, reviews and checklists that apply for the Omega framework are also applicable to all the applications for the development follows similar patterns and it is based on classes and constructs of the Omega. After all this was the reason for which Omega development platform was designed and implemented.
- *Omega Finance*. This was a recently launched project that co-operates with the already developed Omega business module of Accounting. The development of this project was at that time at the Design stage, so the modified PSP could be applied from the very beginning of the implementation. Two senior S/W engineers from the PSP pilot team were involved in this project.

The next step was to modify PSP to address our needs. The modified PSP was designed by taking into account the pros and cons of the PSP as well as the capabilities of the existing Development methodology and tools. The requirements of the modified PSP were as following:

- Introduce as low overhead and disruption as possible.
- Focus on improving time and size predictability.
- Reduce the number of defects found in the delivered modules/products.
- Track and maintain historical data for each Omega project for later evaluation.
- Produce appropriate and complete Design and Code Reviews and Checklists.
- Reduce the overall development time and effort spent on the projects.
- Improve individual's job satisfaction and establish process improvement as an ever-going effort for the benefit of both the developers and the company.

The modified PSP is based on PSP version 1.1 as designed by Watts Humphrey [1] and in the Carnegie Mellon PSP lecture notes [2]. The majority of the scripts and the template forms were introduced in our PSP without any modifications. However, we did eliminate the entries that had to do with Base LOCs, Added, Deleted and Modified LOCs, Object LOCs and Reuse LOCs. We gave the modified PSP the alias of CL_PSP1. For this first version we introduced the Object's classification as described in *Appendix A: Table SME.1*. With this categorisation and with the estimation of the number of objects in the project it is possible to provide estimation for the size and the effort for the project using the PSP PROBE method.

Phase 3: Applying CL_PSP1

The modified PSP, CL_PSP1 was applied in the Omega Development library framework and in the Omega Finance business module. In order to facilitate the tracking of CL_PSP1 data we extended the structure and the functionality of three main utilities used in the Omega Development Process. These are:

- *O To-Do Lists*

This system is responsible for tracking all the development (Analysis, Design, Implementation and Testing) issues that are related to a specific project per developer. The information stored per issue was extended to accommodate the following CL_PSP1 data:

- Estimated Time – Effort (hours)
- Actual Time – Effort (hours)
- Number of defects found in Compilation
- Number of defects found in Unit Testing
- Number of defects found in Integration Testing

- *Work In Progress System (WIPS)*

  This system is responsible for tracking the effort spent by each individual on every day activities that are related to projects or other responsibilities. For the developers involved in the pilot projects (Omega Library and Omega Finance) a new activity has been added called *PSP*, in order to be able to measure the overhead imposed by the introduction of PSP.

- *Requirements Management System*

  This system is responsible for tracking down the Requirements (Analysis and Design) for every project undertaken. This system has been modified in order to accommodate the following CL_PSP1 data (for the Categories of objects used in the CL_PSP1 refer to Table SME.1):

  - Estimated Number of Objects Per Category in the Analysis – Design stage
  - Actual Number of Objects Per Category in the Analysis – Design stage
  - Estimated Number of Objects Per Category in the Implementation stage
  - Actual Number of Objects Per Category in the Implementation stage

The next step was to train all the developers (four S/W Engineers) involved in these projects on the CL_PSP1. The training was based on the curriculum of the PSP1.1 course that the pilot has attended. This first effort of training Omega developers in CL_PSP1 gave us a better insight of the anticipated problems we would face when introducing PSP in all the Omega development projects. The main problem we faced was to convince the developers for the necessity of using PSP. They considered PSP as an effort of the top-level management to pinpoint the way they should work rather than a means to improve their-selves. This form of resistance and scepticism was expected but it was overcome by having the two pilot teams organise a free discussion session in which they explained and shared their impressions and results from the PSP course.

Phase 4: Introducing PSP in MIS Reports

The Reports and Management Information Systems components development process has many common characteristics with the S/W – code development process. The Omega development infrastructure has focused on the improvement of the S/W process. However, the Reporting and Information Processing subsystems (OLAP, Data Drilling, Data Warehouse) on top of the Omega applications have become very important for the overall success of the products being delivered and for achieving high degree of customer satisfaction (Total Solutions).

The productivity of the MIS Reports development process was poor and the quality of the delivered systems was rather low. In order to improve this process we performed two combined actions. The first one was to improve the technological infrastructure (Omega Development platform) and to provide the Report developers with a powerful platform that encapsulates the difficulties of Report writing and facilitates this process. The second action was to experimentally introduce *CL_PSPR1* into the Report development process.

The CL_PSPR1 process is a modified version of CL_PSP1 that has been tailored and customised to address the needs and the individual characteristics of Report development process. This was a rather interesting experiment with very promising results. First of all we had to provide the semantics of the mapping of the S/W Code world to the Report development world. For example the concept of object was mapped to the concept of the Report or Report Element (Cross-tab report, Section Element, Group Element, etc.). The next step was to provide the appropriate categorisation of Report Elements and their estimated effort based on the data we gathered from the Report Development process. The categorisation of Report Elements is described in *Appendix A: Table SME.2*.

Furthermore, we introduced the following Defect Type standard for the defect categorisation in the Report Development process as described in *Table SME.3*.

The action of compilation was mapped to the action of Preview – Run of the Report. Testing remains the same in principle with the CLP_PSP1 (we used the same Test Templates and test forms as in CL_PSP1) since the semantics of testing is to validate that the delivered item performs correctly according to the specifications. Regarding the reviews and the checklists, CL_PSPR1 has also a Design Review and a Design Checklist while instead of Code review it defines a Report Layout Review (Report Layout checklist) and a Database Processing Review (Database Processing checklist).

The report developers were not formally trained to PSP but only to CL_PSPR1 process and to the significance of the introduced metrics. Since CL_PSPR1 is rather a simple and straightforward process with minimal overhead no significant resistance has been identified. On the contrary, the report developers showed enthusiasm and commitment to fill-in and take advantage of the collected data.

Chapter 4: Measured Results and Lessons Learned

The CL_PSP1 was applied in the Omega development Library for a period of four months during which three major versions and three minor versions of the Omega library framework were released. Concerning the Omega Finance project, CL_PSP1 was also applied for the same period of time during which four internal versions (development increments) have been released.

The raw data from enacting CL_PSP1 were processed and interpreted and we reached to the following results:

- Number of defects found in test was reduced by 15%. This was a very promising indication especially for the case of the Omega development library because these effects impact all the Omega applications.
- The average estimation error for well-defined development tasks was measured around +/- 10% which is a significant improvement from what we were used to (+/- 20%). The above measurements exclude the cases where the requirements or the specifications were changed during development. These cases do not meet the assumptions for applying PSP (sound and complete specs) but they often occur in real-life.
- Thorough processing of the collected data revealed that the Estimation – Effort table (Table SME.1) was not accurate enough in the case of the complex UI elements and objects. So we reached the conclusion that we do need to fine-tune these two categories. One possible solution is to split these categories into more so we could cover a wider range of objects or require splitting these complex entities into more manageable and predictable units.
- No actual improvements in the productivity were identified. This was justified by the fact that the period of four months is not enough to justify improvements at the level of individual. We do expect that in one year of systematic use of CL_PSP1 there would be significant improvement in the overall development productivity. We should also take into account that the Omega Development process has been improved in terms of productivity in the context of CMM Level 2 [3] (strong indications of Level 3) (top-down approach) and the ISO 9000-1 assessment. However, the time spent in compilation was reduced and thus we had an indirect productivity improvement (in most of the Omega projects a full-build compilation requires a substantial amount of time).
- The measured overhead of applying CL_PSP1 was on average 15%. However, we do believe that the systematic use of PSP will reduce this overhead to 5% since it will become a natural way of doing the every day tasks.
- Self-discipline and visibility to each individual's progress has been increased. We did observe that the acceptance of the significance of the compilation errors, lead the developers to produce more mature code at the coding stage (prior to the first compilation). Furthermore, they exhibited a self-motivated behaviour to minimise the number of defects found after Unit Testing reducing thus in the long term the effort spent in Integration Testing.

Concerning the Report Development process the CL_PSPR1 was applied for a period of three months in the development of MIS components for the Omega Accounting business module. The results obtained from the enacted CL_PSPR1 data were quite promising:

- The number of defects found in the Test stage was reduced significantly (40 %).
- The effort estimation accuracy was as good as +/- 20 %. This is considered as a significant improvement since prior data indicated a high distribution of the estimation error.
- The overall productivity i.e. the number of MIS components developed per category per time unit, was increased by an average of 15%. This was expected to be higher but we do believe that in the future will be increased even further.
- The overall quality of the delivered components was increased due to the compilation of appropriate Report Review checklists (Design Review, Report Layout Review, and Database Processing Review). The measurement was based on the number of Fix Issues allocated to a Released MIS component. However, this improvement cannot be attributed solely to PSP for during the experiment we had also modified the Report development technology.
- The Report development process was prior to CL_PSPR1 considered as a black box. Tracking of the process was very difficult and even the developers could not evaluate the percentage of the work done. Even worse, it was difficult to estimate how much effort was spent on certain reports. With the pilot introduction of CL_PSPR1 we managed to establish discipline and re-organise the Report development process. Maintaining data on effort, defects, report element types and size made possible the accurate evaluation and therefore better estimation of the cost of the MIS components.
- Increased job satisfaction for the roles involved in the Report Development process. This was very important because the CL_PSPR1 acted as a proof of management attention and demonstrated the significance of the process to the developers.

The results we obtained from applying CL_PSPR1 were very promising and proved that the experiment was successful. The top-level management was convinced about the benefits we would gain from further improvement of CL_PSP_R2 (version 2) and from the introduction of PSP in other development tasks apart from coding.

Conclusions

The experiment of introducing PSP in the Omega production environment has shown promising results and it has indicated ways and practices to improve the company’s S/W Development process.

PSP has been proven to be a quite effective approach in improving the development process from CMM Level 3 to Level 4. There is a significant gap between these levels that has mainly to do with measurements (metrics) and discipline. PSP manages to make the developers believe in the importance of certain measurements thus reducing the resistance and the overhead of performing Level 4 activities. Furthermore, the measurements introduced by PSP are at the right granularity level so that statistical processing would facilitate the control of the process.

Through the experiment it became evident that a repeatable process infrastructure (CMM Level 2) substantially supported the use of PSP. In our case, relatively simple modifications to existing processes and tools provided the means to smoothly enact PSP.

PSP Training in parallel with the every day workload and commitments has presented major obstacles to the developers which were overcome with the exceptional commitment from their part. However, this can not be considered as the normal case so top management commitment and proper scheduling of the training period are required.

The measurements collected from the use of CL_PSP1 and CL_PSPR1 were in general less than what has been set as target. However, the improvements are still significant and they are certainly promising. Furthermore, the benefits from the PSP are manifold indicating that the systematic use of PSP will improve the performance and maturity of the Omega development process at large.

In our case the use of PSP is justified and recommended in the development of core, critical or highly reusable components, where the cost of errors and the need for accurate time estimations is very large. At the current state, top management is convinced about the ROI benefits of applying PSP on the above type of development.

The formal PSP training is not performed at the initial Omega training but rather is provided as an incentive to best performing developers so that they can be used in critical developments. However, the PSP concepts and the CL_PSPxx processes have been included in the Omega training curriculum.

Future actions include the refinement of CL_PSP1 and CL_PSPR1 and the training of more developers in PSP. Furthermore, we plan to introduce PSP in more processes and activities of the Omega infrastructure such as the Customer Implementation Services process.

Having PSP introduced into core processes of the Omega our following process improvement efforts will be targeted for the CMM Level 4.

Appendix A: CL_PSP1 and CL_PSPR1 tables

Object Category	Description	Estimated Effort
CLISUD_SIMPLE	Simple UI component for the handling of high level domain object	2 – 4 hours
CLISUD_MEDIUM	UI component of medium complexity for the handling of high level domain object	4 – 8 hours
CLISUD_COMPLEX	UI component for the handling of two or more high level domain objects or objects with complex interaction with the user	8 – 16 hours
CL_EXPLORER	UI component for the presentation and management of multi-dimensional information in hierarchical layout	1 – 2 hours per actual level
CL_SCROLLER	UI component for the presentation of information in a tabular layout	1 – 4 hours
CLUI_CUSTOM	Custom UI component	2 – 8 hours (depends on the complexity of the controller)
CLDOM_SIMPLE	Domain object of zero – low complexity	1 – 2 hours
CLDOM_MEDIUM	Domain object medium complexity and medium interaction with other objects.	2 – 8 hours (depends on the number of associations to other objects and on business domain)
CLDOM_COMPLEX	Domain object high complexity and heavy interaction with other objects of the system	8 – 32 hours

Table SME.1: *CL_PSP1 Omega Object Categories – Effort Estimation*.

Report Element Category	Description	Estimated Effort
CLR_SIMPLE_DBOBJ	A report based on low complexity database processing	0,5 – 1 hour
CLR_MEDIUM_DBOBJ	A report based on medium complexity database processing	1 – 3 hours
CLR_COMPLEX_DBOBJ	A report based on medium complexity database processing (either large number of database objects or complex queries)	3 – 16 hours (depends on the report's requirements and domain's complexity)
CLR_SUBREPORT_OBJ	A report object that is interfaced to the main report	1 – 2 hours (plus the effort for the sub-report development)
CLR_GRAPH_OBJ	A graphics object included in the report	0,5 – 1 hour
CLR_CROSSTAB_OBJ	A cross-tab object included in the report	1 – 2 hours
CLR_DESIGN_OBJ	All the layout elements that implement the UI of the report	1 – 8 hour (depends on the complexity of the UI requirements of the report)
CLR_GRP_SORT_OBJ	Grouping and sorting aspects of a report	1 – 2 hours

Table SME.2: *CL_PSPR1 Omega Report Element Categories – Effort Estimation*.

Defect ID	**Defect Description**
10	Design object error
20	Design object logical error
30	Database query processing error
40	Database query processing logical error
50	Layout – UI error
60	Report's logic error
70	Parameter error
80	Omega Interface error
90	Configuration system error

Table SME.3: *CL_PSPR1 Defect Types in Report Development Process*.

Defect ID	**Defect Description**
10	Documentation
20	Syntax
30	Build, Package
31	*Configuration Management*
40	Assignment
50	Interface
51	*COM – OLE*
60	Checking
70	Data
80	Function
90	System
100	Environment

Table SME.4: *CL_PSPR1 Defect Types in Omega Development Process*.

Appendix B: PSP course aggregated results

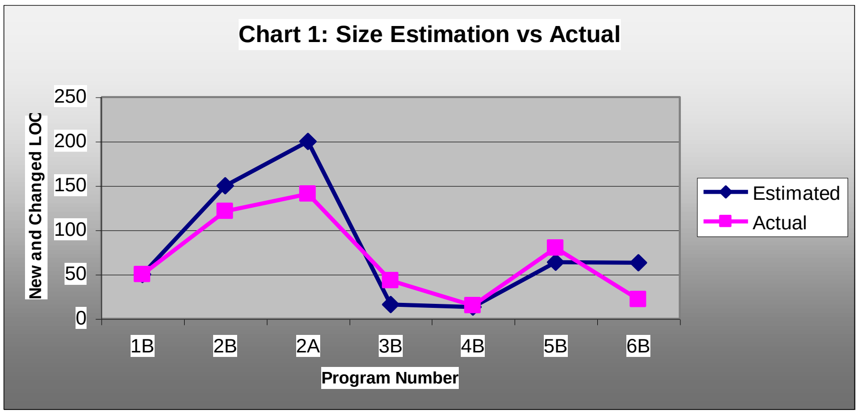


Chart SME.1: Accuracy of the size estimation throughout the course-works. The two lines should be as close as possible increasing thus the predictability of size.

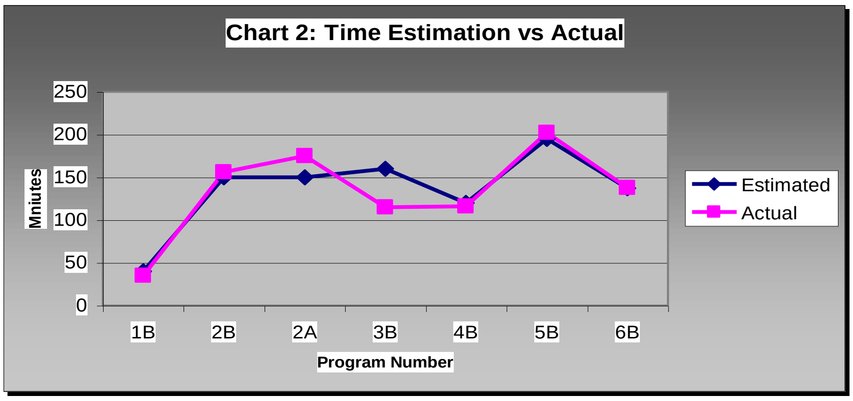


Chart SME.2: Similar to SME.1 with the difference that it demonstrates the accuracy of time estimation.

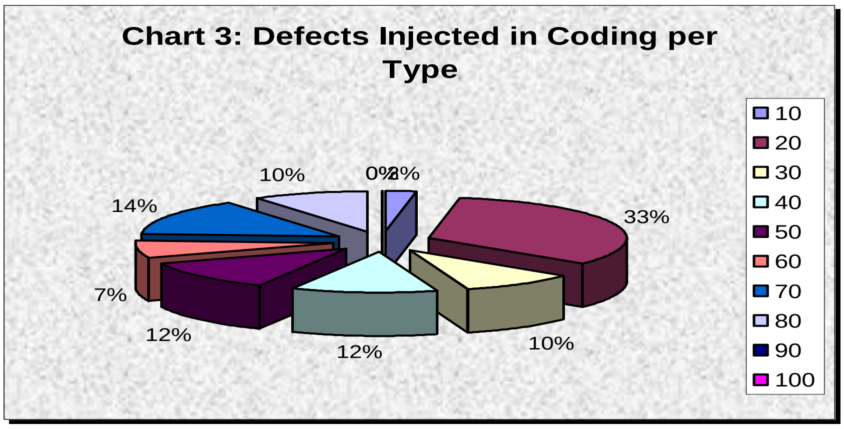


Chart SME.3: The percentage for each type of defect injected in the Coding phase. These data would form the basis for the compilation of the appropriate Code Review Checklists. The PSP 1.1 and CL_PSP1 categorisation of errors is shown in Table SME.4

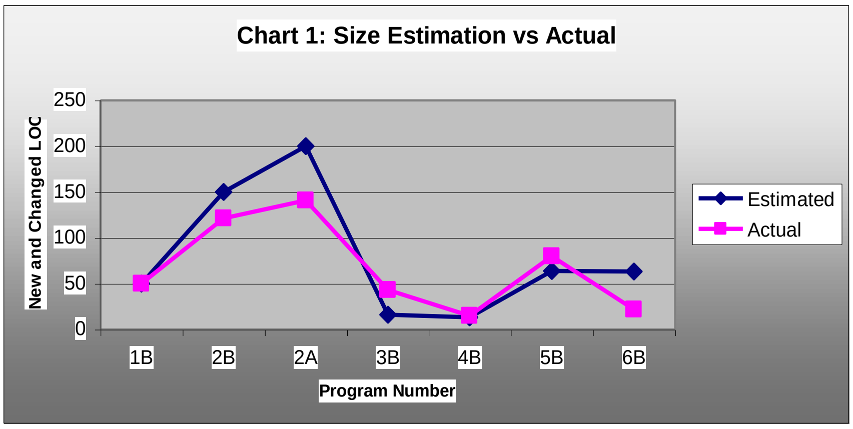


Chart SME.4: The percentage for each type of defect that is removed in the Test phase. These data would guide the construction of the most appropriate Test cases instructions for the testers. The feedback provided to the developer helps him to understand to which type of errors is more vulnerable.

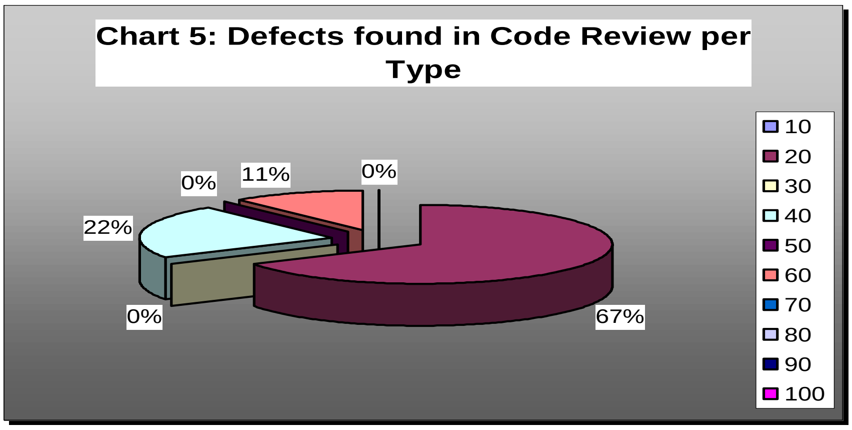


Chart SME.5: The distribution of the defects found in Code Review per defect type. This information along with the data from Chart 4 form the guide lines for the Code Review Checklists.

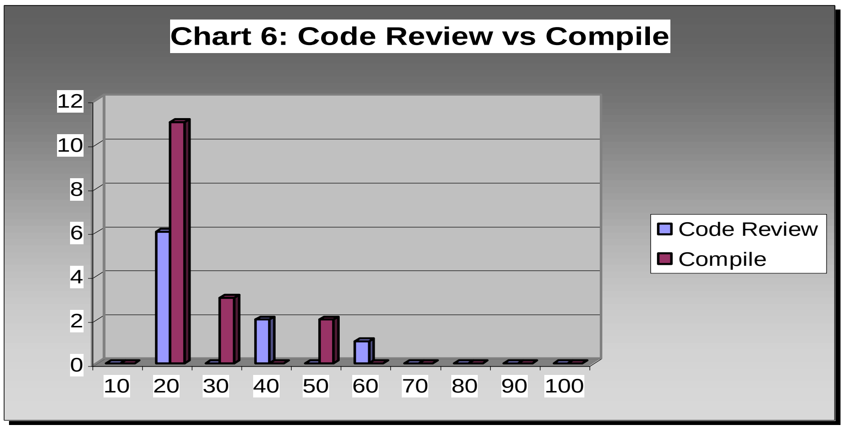


Chart SME.6: Comparison of defects found in Code Review versus Compile phase. We should try to maximise the number of defects found in Code Review and minimise those found in Compile phase.

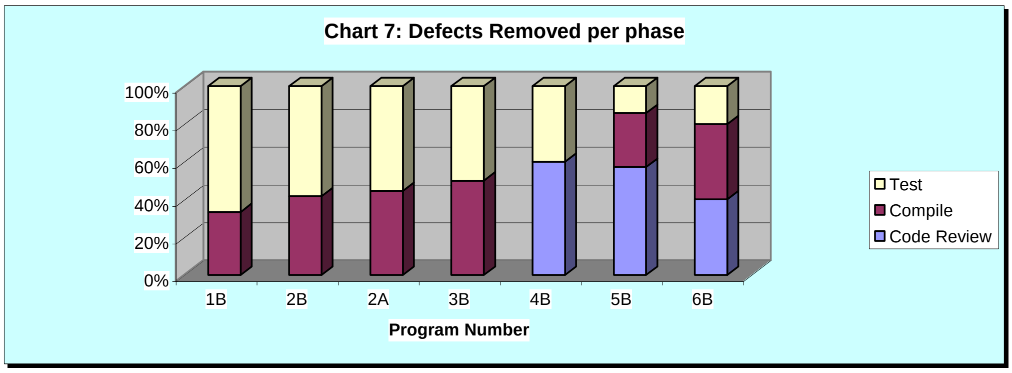


Chart SME.7: Percentage of defects found in each phase. The ultimate goal of PSP and therefore CL_PSP1 is to minimise the number of defects found in Test because they are more expensive to fix.

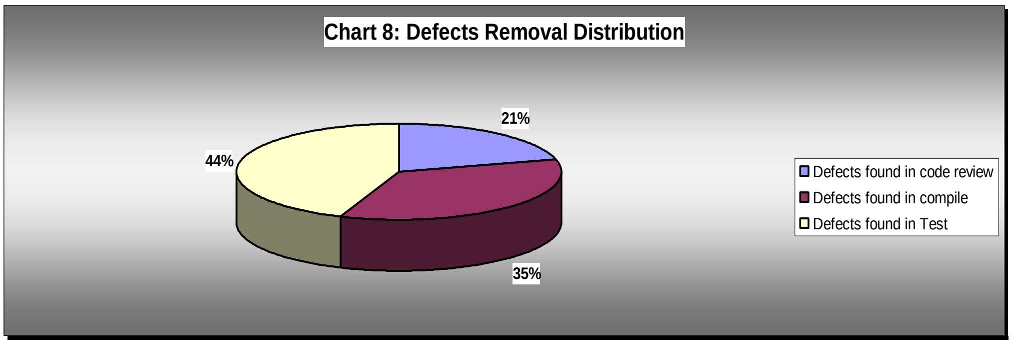


Chart SME.8: The percentage of defects removed in each phase. Similarly, PSP's goal is to remove defects in the early stages of development.

References

[1] Watts S. Humphrey, *A Discipline for Software Engineering*, Addison Wesley.

[2] Carnegie Mellon University, *Personal Software Process*, *lecture notes*

[3] Avratoglou Ch., *Process Management Improvement for Software Development*, *ESSI Project: 21551*, Final Report, 25 Aug 1997 (http://www.esi.es/VASIE/Reports/All/21551/Report/index.htm)

Session 12 – SPI Processes and Modeling

Integrated Requirements Engineering Approach for Systems and Software

Ulrich Zanker
Liebherr-Aerospace Lindenberg, Germany

Modelling Guidelines for Outsourcing Projects

Christian Zwanzig
ATB Institute for Applied Systems Technology Bremen GmbH, Bremen (Germany)

CCM - A fundamental Process for Improving Quality

Clemens Gasser
Joanneum Research, Graz, Austria
Edwin Deutschl
Joanneum Research, Graz, Austria.

Integrated Requirements Engineering Approach for Systems and Software

Ulrich Zanker
Liebherr-Aerospace Lindenberg, Germany

Introduction

Executive Summary

This paper describes the process improvement experiment conducted on a helicopter fly-by-wire flight control project.
A typical development project is sequenced in the following main activities:
***System Requirements Analysis -> Software Requirements Analysis** -> Software Design -> Software Coding -> Software Verification.*
The process activity to be improved is in the area of system/software requirements analysis and capturing. To reach such an improvement, the idea was to use the dynamic system modelling technique for the behaviour system model (System Requirements Analysis) and directly link mathematical performance simulation models for early specification validation and rapid prototyping.
The new requirements engineering approach integrates all electronic design disciplines, system, software, hardware and quality assurance. Benefits for the software design process is a direct target and to establish a solid basement for further process improvements (i.e. hardware/software co-design) is the expected positive long term spin-off.
The original goals already achieved are the selection of an appropriate tool, training of the involved engineers, implementation and settlement of the tool and method within the system/software process. An initial package of the baseline projects specification is captured, validated by simulation and transformed to software requirements. These initial software requirements are rapidly prototyped and the risk areas are checked. A second run through system and software requirements analysis and capture was performed (enhanced application software functionality). Currently the documentation generation, result analysis and measurement and dissemination activities are in progress. The lessons learned are:

- A significant effort is necessary to select a tool/method to fit both, the systems and the software engineers needs and to be appropriate for the application to be developed.
- Having such a tool/method in place and disciplines/engineers working close together for specification validation is very beneficial in terms of selecting the appropriate system architecture.
- Specification validation and rapid prototyping gives a good confidence in the selected hardware/software architecture at an earlier stage than with conventional projects.
- A further early result is the fact that the automatic code generation capability, provided by the simulation tools, are could not be used for this project due to the selected „unique" hardware/software design. This leads to the expectation that automatic code generation cannot contribute a significant benefit to this type of application (dedicated, unique airborne equipment), but could have an enormous potential when using standard (off-the-shelf) equipment, which is a clear trend in the avionics community.
- Tool having excellent simulation capabilities, can be quite poor in documentation generation.

For future projects it can be assumed that the integrated engineering approach will be conducted. The selected method and tool chain will be improved even beyond this PIE project objectives.
The details of this report may be of a particular interest for the aerospace community but also for organisations producing embedded system software using none-standard equipment.
This project is carried out by the electronic division of Liebherr-Aerospace Lindenberg with financial support from the European Commission under European Software and Systems Initiative ESSI, Software Best Practice, Process Improvement Experiment PIE.

Article Content

- a short description of the company / business / product
- the starting scenario
- the plans and the expected outcome
- the implementation of the improvement actions
- the measured results and the lessons learned

A Short Description of the Company/Business/Product

LIEBHERR-AEROSPACE LINDENBERG GmbH, part of the LIEBHERR international group, works with about 1200 employees exclusively in the aerospace business, being one of the leading European equipment suppliers for civil and military aircraft's.
The product spectrum of LIEBHERR-AEROSPACE LINDENBERG GmbH is diverted into the following areas:

- primary and secondary flight controls actuation systems
- landing gears
- environmental control systems
- embedded electronics

This products are supplied to aircraft- and helicopter manufacturers like: Airbus, Dornier, Embraer, IPTN, Canadair, IAI, Eurocopter, Boeing, etc...
LIEBHERR-AEROSPACE LINDENBERG is regarding itself as a system rather than a component supplier. This causes the demand to deal with control applications, which is electronics in general and software in particular. About 100 employees are dedicated to electronics development and production, 15 of them are in charge with software development and test for advanced fly-by-wire systems, environmental control systems and actuator embedded electronics applications.
The importance of software within these systems is steadily increasing, but also the costs of developing and maintaining software has increased to a level hardly tolerable. Nevertheless there is no way for a system supplier to omit electronics and software development, while keeping its position within this highly competitive global market. Assessments have shown that the only way to manage the situation is an incremental but continuous improvement of the development process.
The establishment of a solid basis (advanced requirements engineering methods) for this improvement activity is manpower, time and money consuming and risky in terms of immediate success. The european process improvement experiment (PIE) has been an opportunity to lower these risky items (in context with a real world program) to manageable values.

The Starting Scenario

The system and software development process for aircraft equipment is strictly regulated by advisory papers like ARP4754, DO-178B or DOD-STD 2167A. In order to be compliant with those papers a V-Model software development process is established at LIEBHERR.

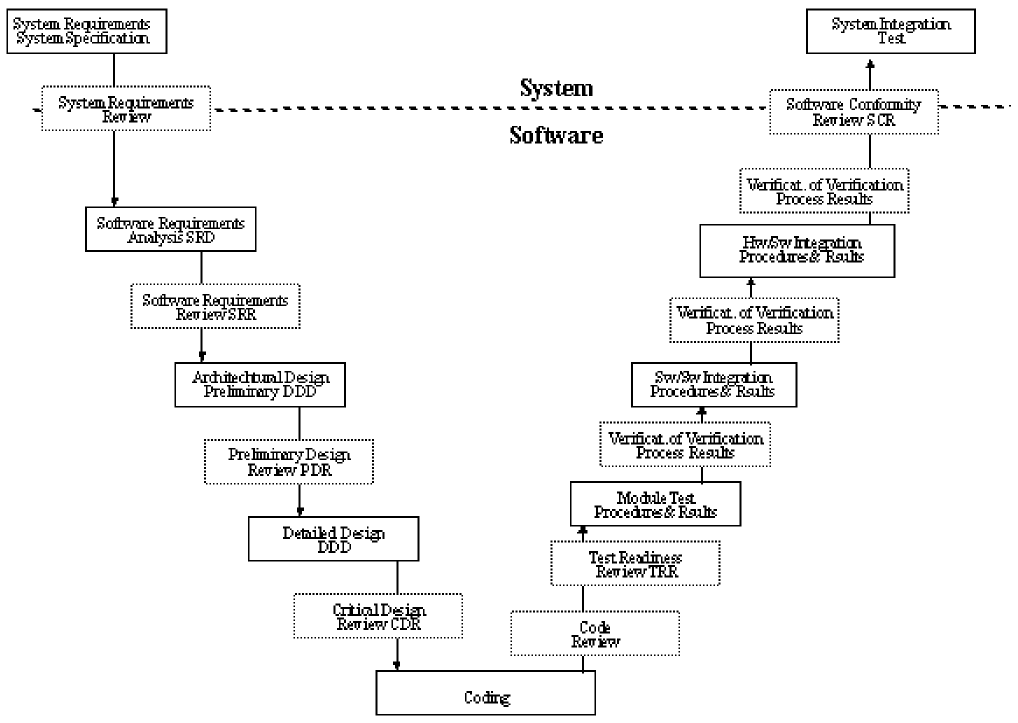

System Requirements System Specification
System Requirements Review
System
Software
Software Requirements Analysis SRD
Software Requirements Review SRR
Architectural Design Preliminary DDD
Preliminary Design Review PDR
Detailed Design DDD
Critical Design Review CDR
Coding
Code Review
Test Readiness Review TRR
Module Test Procedures & Results
Verificat. of Verification Process Results
Sw/Sw Integration Procedures & Results
Verificat. of Verification Process Results
Hw/Sw Integration Procedures & Results
Verificat. of Verification Process Results
Software Conformity Review SCR
System Integration Test

Fig. IREASS.1 : Software Lifecycle

System Requirements Analysis

The customer requirements are analysed and a system specification is developed, by making use of simulation tools (Matlab) and system design experience. The results of the analysis is documented and made available by pure paper output (specification). This specification is now used by the special departments, to derive system test cases, test equipment requirements, hardware requirements and software requirements.
The strengths of this proceeding is in having no formal or method overhead, design engineers work with best effort dependant on experience only.
The weak points are in having no general method, isolated tool data (simulation), communication problems with other system designers and special departments, no consistency analysis, no support for test case generation, no database export for sub-specifications.
The envisaged corrective actions is the establishment of an state of the art requirements engineering method with integrated tool support.

Software Requirements Analysis

The system requirements are analyses and the software requirements are developed, by making use CASE-tool supported structured analysis (SA) and real time analysis (RT). The results of the analysis is documented and made available with the software requirements document. This software requirements are now used to derive hardware/software integration test cases and the software design.
The strengths of this proceeding is in having structured method, tools constancy checking capabilities, tools database can be exported to software design process and software testing process.
The weak points are in having no database import from system process, communication problem with system designers, uncovered specification inconsistencies to be fed into time consuming specification change loop, hardware and software development are separated, no support for test case generation, documentation generation semi-automated instead of fully automated.
The envisaged corrective actions is the establishment of an state of the art requirements engineering method with integrated tool support, tool supported test case and documentation generation.

Software Design

The software requirements forms the basis for software design work, making use of CASE-tool supported structured design (SD). The two-phase software design, structure charts (architecture) and module specifications (software unit detail), is fully embedded in CASE-tool environment. The results of the software design is documented and made available with the design description document, now used to develop software integration test cases, software unit test cases and the software code.
The strengths of this proceeding is in having structured method, tools constancy checking capabilities, tools database can be exported to software coding process and software testing process.
The weak points are in having no support for test case generation, documentation generation semi-automated instead of fully automated, limited code generation capabilities.
The envisaged corrective actions is the implementation of tool supported test case definition, enhanced code- and automated documentation -generation.

Software Coding

The software design forms the basis for coding work, using standard cross development tools. C-compilers, host based simulators, ROM debuggers and emulators are integrated within an effective Unix workstation cluster. The software structure is translated to C-code via a code generator while the detailed design is translated manually.
The strengths of this proceeding is in having an effective and integrated cross development tool environment.
The weak points is having very limited code generation capabilities.
The envisaged corrective actions is the gradual upgrade of tool environment to enhanced code generation.

Software Verification

The software verification process consists of reviews, analysis and tests in terms of software unit (modules), software integration (design) and hardware software integration (requirements) verification activities. The depth and the effort of this activities depends on software criticality category. The software verification process activities are documented and made available with appropriate plans, procedures and results.
The strengths of this proceeding is in having review and analysis techniques are supported by CASE environment and requirements traceability tool, statical software unit tests are automated.
The weak points are in having no sophisticated tool support for dynamic unit test, sw integration test and hardware/software integration test, documentation generation semi-automated instead of fully automated.
The envisaged corrective actions is the tool supported software test and automated documentation generation.

Conclusion

With the software fault trend analysis (reasons for software changes), undertaken for a fly-by-wire flight controls project, can be shown that the areas, „customer request“ and „project requirements“ have a proportional high share in the reasons for software changes. The majority of these changes can be dedicated to inconsistencies or white spots of the specification and to communication problems between system and software development disciplines.

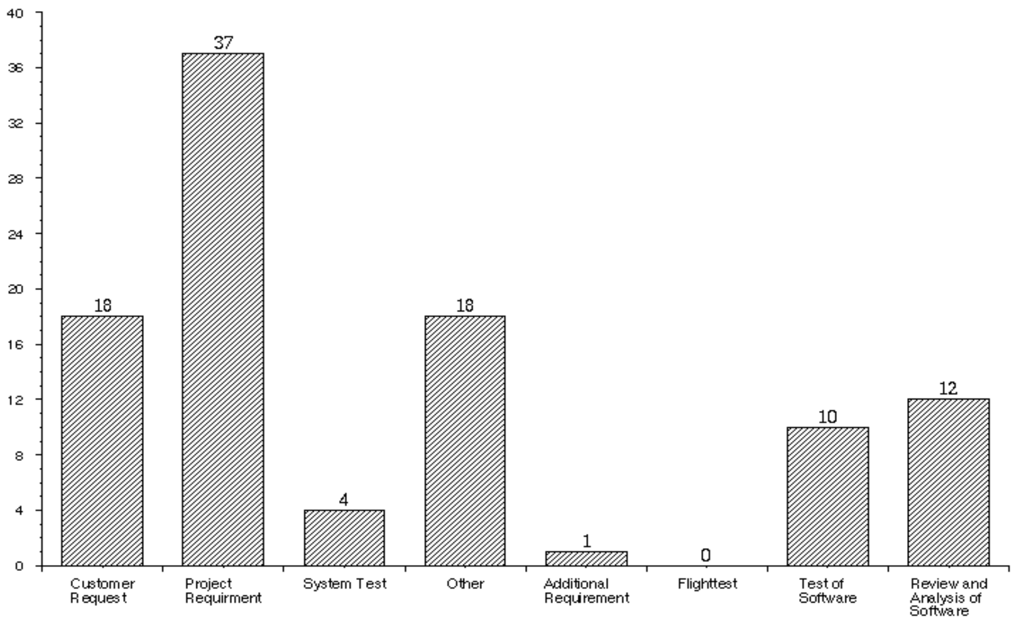


Fig. IREASS.2 : Software Fault Trend Analysis

This lead to the conclusion that the utilisation of suitable methods and CASE-tools for the system specification work would result in best benefits for the development process by founding the solid basement for further process improvement. Such the corrective action encountered above can be prioritised:

- establishment of a requirements engineering method with integrated tool support (Priority 1).
- hardware/software Co-design (Priority 2)
- tool supported software test (Priority 3)
- automated documentation generation (Priority 4)
- enhanced code generation (Priority 5)

The process improvement activities at LIEBHERR are considered as important and essential for the company and its products. The processes are subject to continuous analysis and from today knowledge, the above indicated corrective actions will be released step by step, following the prioritising.

Quality Management

Quality assurance activities are involved in the development process from start of a project to the end of a systems useful life and plays an active role in any process definition/improvement activity. The involvement of the quality assurance organisation, into development and manufacturing processes, is laid down in the company quality manual. LIEBHERR is certified to AQAP-1 and the quality system is compliant to ISO 9001. Compliance to ISO 9000/3 (Software) is a short term target, while certification to ISO 900X is planned for some date in the future.

The Implementation of the Improvement Action

Introduction

The experiments main focus is the definition and introduction of dynamic system modelling methods and techniques with integrated tool support. The selection of the appropriate method and tools has marked the starting point of the experiment, followed by an intensive training suite for the experiment involved system and software engineers. The experiment is finished when having evaluated the results by comparing the experiment overlaid sub-project (ACE) with the reference project(COS).
This PIE-project is completely managed and carried out by Liebherr, the prime user. For training and coaching (consultant) Scientific Computer was selected as a subcontractor, the tool producing company.
The requirements for the actuator control electronic (ACE) shall be developed in conjunction with the process improvement experiment, while the cockpit interface(COS) will serve as a reference project, with a conventional requirements engineering approach. The baseline projects are split into two work-packages each, the „initial package“ implementing the minimum functionality required to make the whole system run and the „final package“ implementing full functionality.
The workpackages (WP 1 - 13) defined below, are in relationship to those „initial“ and „final packages“ and introduced around requirements analysis, capture, simulation and validation. The termination of those workpackages are marked by internal deliverables and internal dissemination activities as defined with the workpackages. This marks serve as measured reference points for the project

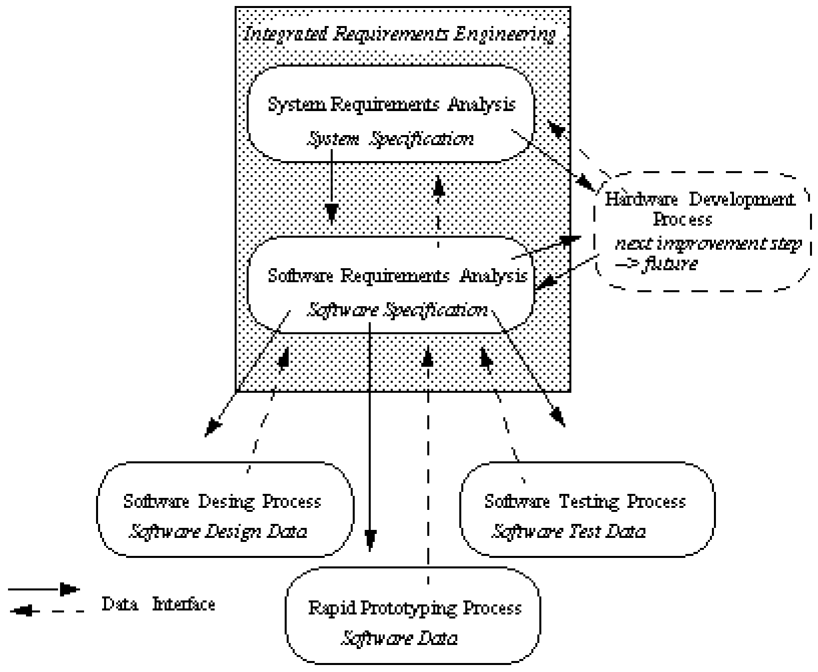


Fig. IREASS.3 : Integrated Tool Overview

The Selected Project

Within the baseline project an electronic flight control system is to be developed for the existing EC 135 helicopter. This program contains two kinds of electronic boxes, a cockpit interface COS and a actuator control electronic ACE. A general system overview is given in figure below:

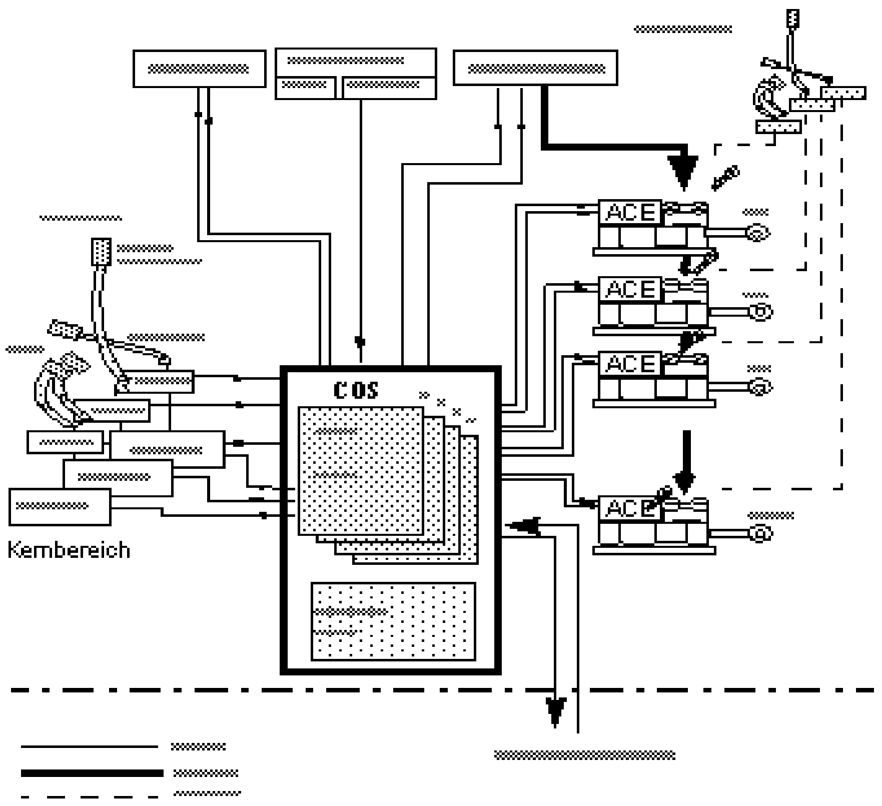


Fig. IREASS.4 : System Overview

Reference Project Cockpit Interface (COS)

Specification of functional requirements, hardware and software requirements, software development, software test and system integration are the activities to be accomplished for that sub-project. The COS is designed to receive and process data from sensors and upstream flight control computers. The resulting commands are transmitted via fibre distributed data buses to the actuator control electronics. In addition the COS has to collect the actual data from the actuator electronics and transmit that data to the cockpit for indication and analysis purposes.

Baseline Project Actuator Control Electronic (ACE)

Specification of functional requirements, hardware and software requirements, software development, software test and system integration are the activities to be accomplished for that sub-project. The functional requirements, laid down within the customer specification, have to be analyses and subsequent system requirements have to be established, captured and validated, in the context of integrated tool support from requirements analysis, via the software design, coding, integration to verification and test. The ACE is designed to receive command values from the COS, closing the actuator regulation loop by calculating the servo-drive current, monitor hardware and behaviour of the connected actuator and transmit status and monitor data back to the COS.

Sub-Projects	Date&Time
COS initial package	mid 97 - mid 98
COS final package	end 98 - mid 98
ACE initial package	mid 97 - mid 98
ACE final package	End 98 - mid 98

Method and Tool Selection WP 1

Selection of appropriate method and tool for the requirements engineering task. The methods and tools were finally analysed for adequacy. The result was the selection of the Matlab/Simulink/Stateflow tool chain.

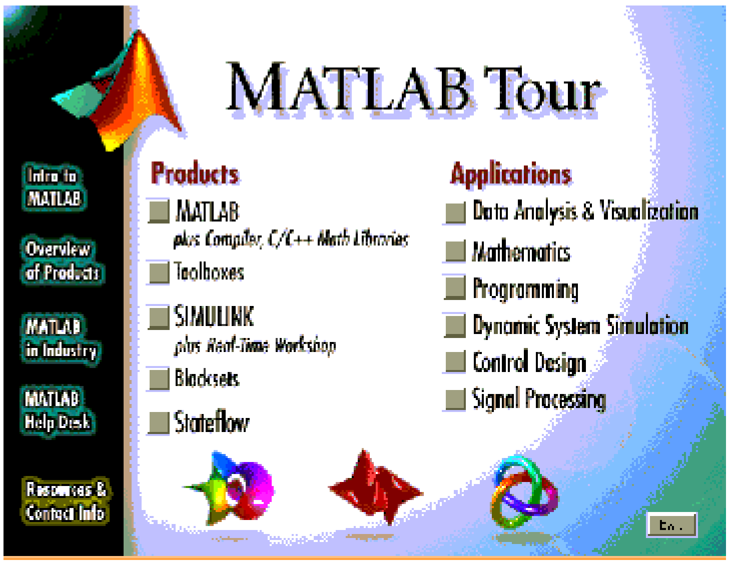


Fig. IREASS.5 : Selected Tool

Basic Training and Tool Installation WP 2

Basic training of method and tool. Engineers of the different disciplines (system design, software design, hardware design and quality assurance which are contributing to the PIE) have been trained together. This was the first step towards the integrated approach.

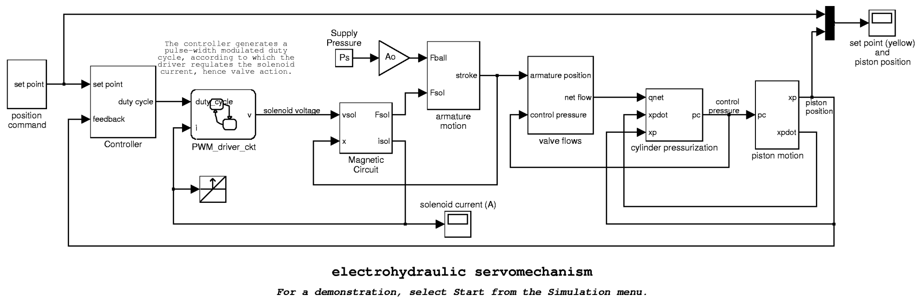


Fig. IREASS.6 : Basic Training (Example Model)

Add On Training WP 3

Sophisticated add on training of method and tool. The engineers of the PIE core team (system design and software design) have been trained together. The aim of this training was to enable the engineers to start a project by utilising system modelling methods and technologies.

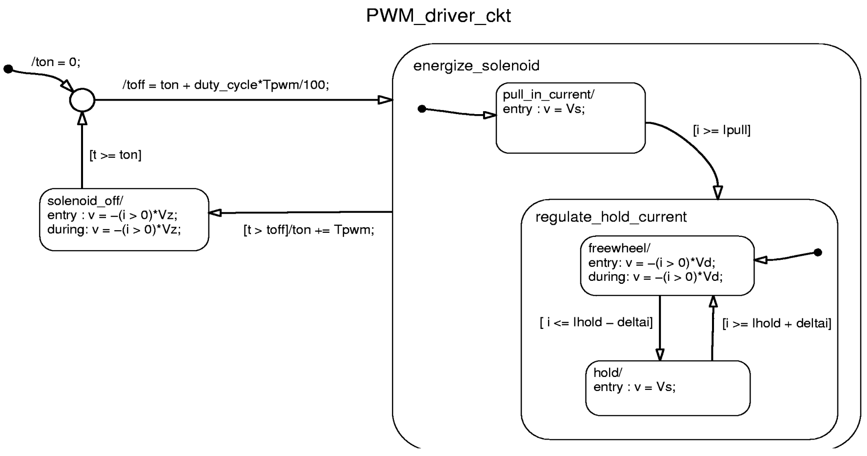


Fig. IREASS.7 : Add On Training (Example Model)

Familiarisation with Method and Tool (Example Application) WP 4

A small example application was modelled from system specification to the software requirements. This has given the chance to familiarise with method and tool.

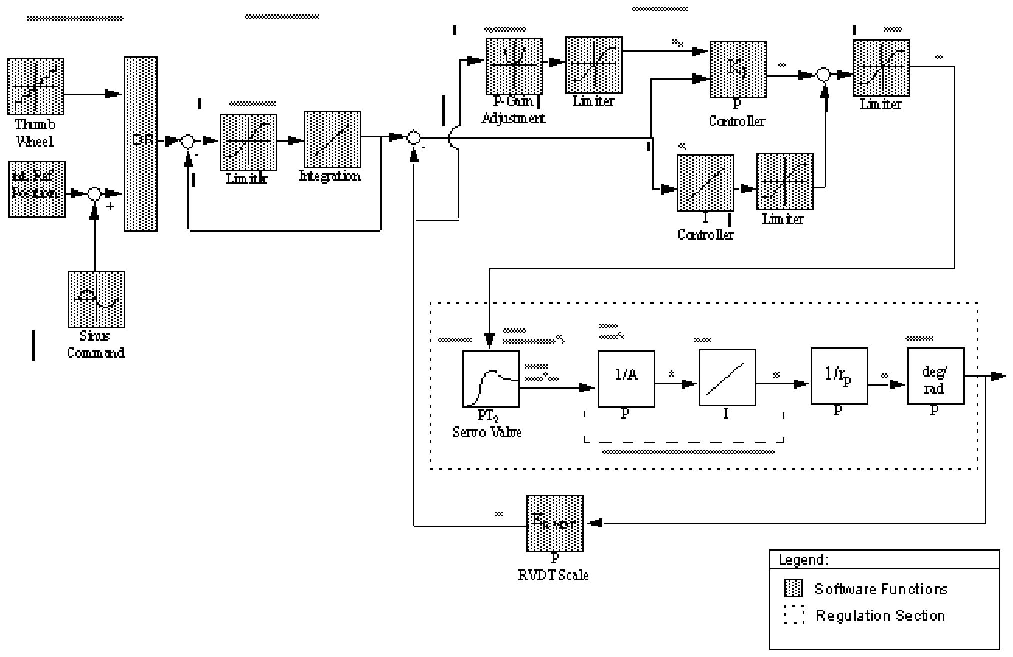


Fig. IREASS.8 : Example Application (Control Loop Model)

System Requirements Capture „initial package“ WP 5

The system requirements for the baseline projects, already formulated with conventional specifications, have been reworked by utilising the systematic methods of dynamic system modelling and the modelling tool support to implement the minimum functionality required to make the whole system run („initial package“). An activity requiring a co-ordination of all design disciplines (system, software, hardware and quality assurance).

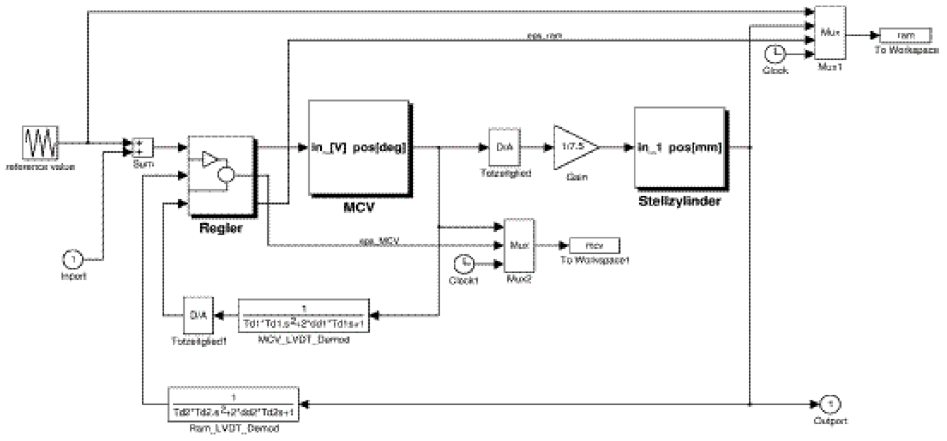


Fig. IREASS.9 : Baseline Application (Control Loop Model)

System Simulation and Specification Validation „initial package“ WP 6

The system requirements for the baseline projects („initial package“) have been validated by analysis and simulation. This activity was concentrated on system design personal (system engineers) but software engineers and coaching support have been involved to reach the goal of an integrated requirements engineering approach.

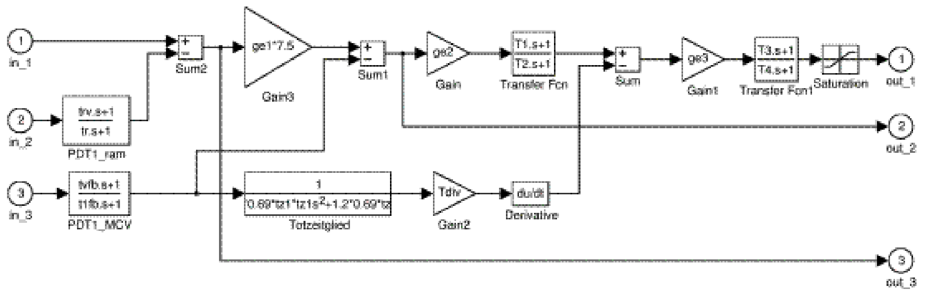


Fig. IREASS.10 : Control Algorithm, Simplex Simulation Model

Export of System Model Database and Software Requirements Capture „initial package“ WP 7

The system models database has been exported to the software process, suitable for refinement of the software requirements from the system requirements. This activity was accomplished by the software engineers actively supported by the system engineers.

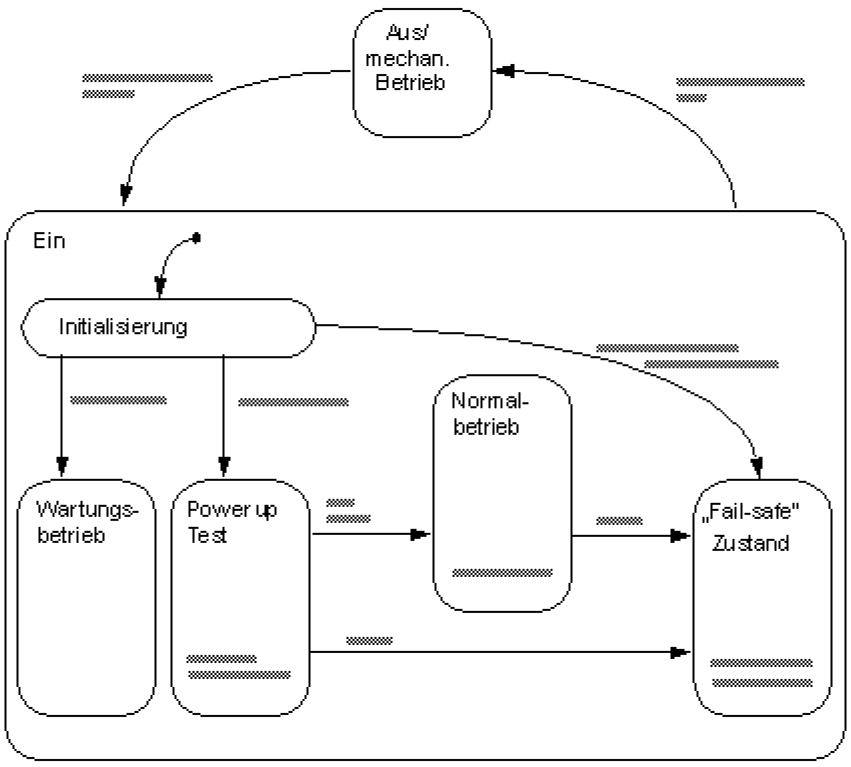


Fig. IREASS.11 : Example Application (Mode Control)

Rapid Prototyping „initial package“ WP 8

Rapid prototyping via automated code generation should have been used to validate the sensitive edges (control loop) of the system/software design, an activity intended to deal with risk management involving mainly software engineers.

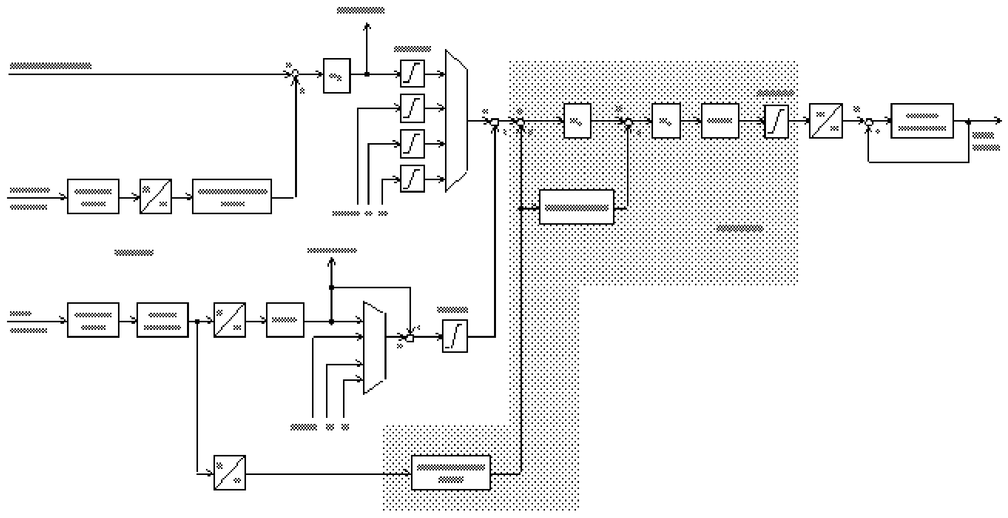


Fig. IREASS.12 : Quadruplex Voting Control Loop

System and Software Requirements Capture „final package“ WP 9

The system requirements for the baseline projects, are reworked by utilising the systematic methods of dynamic system modelling and the modelling tool support to implement the full functionality („final package“). The „ final package“ has to be validated by analysis and simulation. After export of suitable data to software development, the software requirements for the full functionality is refined and fed to the software design process. An activity requiring a co-ordination of system and software engineers.

Documentation Generation for System and Software Specification WP 10

Automated documentation generation of system specification and software requirements document This activity is accomplished by the software engineers actively supported by the coaching experts.

Analysis and Measurement of Results WP 11

Analysis of measured result and comparison with reference project.

External and Internal Dissemination Activities WP 12

Preparation of report and presentation material and subsequent presentation for the intended audience.

Project Management WP 13

Preparation of PIE-project schedules, observation of effort and results and applicability of measured reference points. The project schedules and resources are co-ordinated with the companies product planning system PPS. The technical co-ordination and observation is accomplished by an assigned engineer, assisted by the integrated configuration management and problem reporting tool STS.

The Measurement Results and the Lessons Learned

At this time, the experiment is ongoing and especially the results and analysis are of preliminary nature and not complete. The full and final results will not become apparent until the remainder of the experiment is completed.

Measurement of Results

The measurement of results in form a fault trend analysis is in progress. As a preliminary result we can see good stability of system/software requirements for the ACE control loop (baseline project), whilst the requirements of the COS are still moving.
We expect that the early detection of the ACE functional architecture inadequacy has already contributed positively to the return on investment but this is difficult to quantify. The exceptional "early" stability of the requirements we can observe let us expect positive results from the fault trend analysis and a better lines of code per hour relationship compared to the reference projects measurement.

Lessons Learned

Technical

Tool Selection: - The CASE/Simulation tools available on the market fitting best for standard (off-the-shelf) hardware/software architectures. To select a specification tool for a dedicated and unique application, as it is currently the case in the aircraft actuation and control business, is a complex and sensitive task. Finally a mathematical performance simulation tool (Matlab/Simulink) with its extensions Stateflow and Real-Time Workshop was selected. This tool chain is highly integrated, fits best for inter-discipline (system, software, hardware) specification task, offers an adequate rapid prototyping capability and has, from our point of view, an enormous potential for future extensions. The weak point of the selected tool chain is the documentation feature which is quite poor in comparison with classical CASE tools, but announcements from the suppliers are in place to improve this in the near future.

Training and Familiarisation with Method and Tool: - After the intensive training, the core part of the system specification of a small sample application (landing gear steering system) have been validated. The positive effect, compared to a none PIE overlaid project, was a very detailed knowledge of the steering control problems and side-effects, despite the application is completely new and no prototype hardware is available at that time.

System Requirements Capture and Specification Validation: - The intensive simulation activity for specification validation, driven mainly by the PIE-activity, has finally shown that the selected hardware/software architecture (control-monitor philosophy) is not adequate for this actuator control application. This inadequacy was detected at an very early stage of the project, compared to conventional projects, where the problem would have been discovered much later, probably during system integration phase.

Software Requirements Capture and Rapid Prototyping: - The software requirements capturing activity is considered to be more effective due to involvement of software engineers in specification validation and due to increased confidence into correctness of specified functions and algorithms. The rapid prototyping (within a target prototype) is considered as a second key activity (beside the specification validation). This activity has to deliver the final confidence into the selected hardware/software design to be adequate in terms of performance. An early result is the fact that the automatic code generation capability, provided by the simulation tools, are could not be used for this project due to the selected „unique“ hardware/software design.
This leads to the expectation that automatic code generation cannot contribute a significant benefit to this type of application (dedicated, unique airborne equipment), but could have an enormous potential when using standard (off-the-shelf) equipment, which is a clear trend in the avionics community.

Documentation Generation: - The documentation generation capability is currently an extremely weak point of the selected method/tool. In that area we are missing the automated assistance of a "traditional" CASE tool. The documentation generation for the project is not automated as expected.

Business

It is to early to describe results in terms of their business impacts definitely, but as an outlook, the assumption of the program plan, that a development process, effective in terms of cost and time to market is essential, is still valid and the selected tool/method has the potential to deliver this.

Organisation

The project had no direct impact on the organisational environment, but the project results clearly confirms and support of the recent trend within the company to establish a product related organisation network, in addition to the pure departmental structure of the past.

Culture

In general there are mainly positive impacts related to people. The different design disciplines improved there communication basis dramatically. The early involvement of software engineers in specification related activities and the get in touch of the system engineers with software related problems can both be considered positive.

The natural resistance of our system engineers to do their daily business by using complex tools and methods led to the selection of a tool chain fitting both, solving real world practical problems and the more academic philosophy of the software staff. The final success, discovering design inadequacy at this early stage and having solid requirements proved by rapid prototyping, already convinced the two parties having done the right selection.

Skills

The additional skills learned from training and practising wit the PIE-project gives a broader knowledge the formerly specialised experts. Especially the system and software designers are found together to an integrated team with better understanding of each others problems which is beneficial for the entire product.

Key Lessons

For the time being, the key lessons cannot be complete neither have the final format but first lessons already learnt and formulated.

Technological Point of View

A tool chain which is expected to fit for requirements engineering and enhance productivity needs to:

- have an excellent user interface provide good mathematical performance simulation capabilities
- have behavioral simulation capabilities
- provide automatic code generationhave standard interface to other tools (i.e. prototyping tools, configuration management, etc..)
- provide state of the art documentation capabilityhave future extension potential

There is no such tool on the market. You have to compromise, set priorities.

The use of standard tools is most effective if using standard hardware/software architectures/platforms and less effective when working with dedicated, unique designs.

Early specification validation is a key feature for adequate system, hardware and software design.

Rapid prototyping, with target prototypes is essential for design confidence in case of embedded real time applications.

Business Point of View

There is a definitive need of continuos improvement of the development process in terms of cost effectiveness and time to market.
The market provides new tools, technology and knowledge within very short update rates, but not every technology or tool fits to a given application.
There is an excellent return of investment if choosing an adequate tool and method and there is only cost if selecting an inadequate.

Strength and Weakness of the Experiment

Despite the analysis and measurement activity is not completed, a preliminary view on experiments pros and cons can be made.
Pros:

- having selected good tool and method for integrated approach
- intensive specification validation show benefits as expected
- integrated approach shows positive results (communication, problem understanding, etc.)
- adequate specification process is a good basis for further improvements- the experiments theory is continually proofed by the baseline projects real world problems

Cons:
- compromises at tool selection for software development (simulation tool - not CASE tool)
- finding an integrated approach (system, software, hardware) can be costly in time and money
- the experiments success may be influenced by baseline project success

Conclusions and Future Actions

The experiment is ongoing and especially the results and analysis are of preliminary nature and not complete. The full and final conclusions will not become apparent until the remainder of the experiment is completed.
As a first and preliminary conclusion the conduction of the experiment has shown the correctness and adequacy of experiments philosophy - founding a solid basement with the integrated requirements approach for future process improvements. It became obvious that process improvement has to be a continuos action driven by the appearance of new technologies.
A key feature to gain good support from the market (tools, expert knowledge) is to change to standard architectures in software and hardware as far as it is possible in our field of applications.
It is evident that we have to apply the selected simulation environment to a new project, having in mind the strength and weakness encountered during the experiment. And we shall try to introduce the next step, identified when analysing the starting scenario - the hardware/software co-design.

References

[1] RTCA/DO-178B, Software Consideration in Airborne Systems and Equipment Certification, RTCA-Requirements and Technical Concepts for Aviation, RTCA, Inc. 1140 Connecticut Avenue, N. W., Suite 1020 Washington, D.C: 200036, USA 1992

[2] ARP 4754, Certification Considerations for Highly-Integrated or Complex Aircraft Systems, System Integration Requirements Task Group, SAE-Society of Automotive Engineers, Inc., USA

[3] DOD-STD-2167A, Defense System Software Development, Military Standard, Department of Defense Washington D.C. 20301, USA 1988

Glossary

ACE	Actuator Control Electronic
ACT/FHS	Active Control Technologie Demonstrator - Fliegender Hubschraubersimulator
CASE	Computer Aided Software Engineering
COS	Cockpit Interface
DLR	Deutsche Forschungsanstalt für Luft- und Raumfahrt
ECD	Eurocopter Deutschland
LLI	Liebherr-Aerospace Lindenberg GmbH
PIE	Process Improvement Experiment
PPS	Product Planning System
SA	Structured Analysis
SD	Structured Design
RT	Real Time

Appendix

Author

Ulrich Zanker, graduated at Fachhochschule Munich with Dipl. Ing. (FH), is working for Liebherr-Aerospace Lindenberg since 1983.
He has experience in developing software for airborne computers like: Airbus A310 Slat Flap Control Computers (Flight Controls) and Airbus A320 Zone Controller (Environmental Control System).
Another field of experience is the introduction and administration of the technical computer system for the electronic department (VAX-Cluster, Unix Workstations and networked PCs).
Since 1993 he is heading the software group (currently 16 software engineers) of the Liebherr-Aerospace electronic department, producing safety critical software for airborne computers as well as software for the associated test systems (test benches and simulators).

Company

LIEBHERR-AEROSPACE LINDENBERG GmbH, part of the LIEBHERR international group, works with about 1200 employees exclusively in the aerospace business, being one of the leading European equipment suppliers for civil and military aircraft's.
The product spectrum of LIEBHERR-AEROSPACE LINDENBERG GmbH is diverted into the following areas:

- primary and secondary flight controls actuation systems
- landing gears
- environmental control systems
- embedded electronics

this products are supplied to aircraft- and helicopter manufacturers like: Airbus, Dornier, Embraer, IPTN, Canadair, IAI, Eurocopter, Boeing, etc..

Modelling Guidelines for Outsourcing Projects

Christian Zwanzig
ATB Institute for Applied Systems Technology Bremen GmbH, Bremen (Germany)

Introduction

Project Background

A lot of commercial and industrial organisations need and use application-specific software systems in order to fulfil their daily business according to market requirements and in order to gain, save and expand their market shares. The necessity for application-specific software systems will even dramatically increase in the future because of the steadily increasing international competitive pressure apparent in the growing market demands placed on cost-efficiency, quality and flexibility with respect to services and products. The development of application-specific software needed for running daily business is on the one hand often independently executed by these organisations. On the other hand, organisations due to the lack of required developmental resources have to use external software suppliers to fulfil this task.
Extensive experience has shown that it is often much more efficient (i.e. cost and time aspects) not to develop software systems fully in-house, even for commercial and industrial organisations with their own software development departments. However, being aware of the additional development risks connected with outsourcing, organisations still are reluctant to extensively use this possibility. Thus, the advantages of a short time to market period and the cost efficiency of a specialised software supplier are not fully exploited.
A key factor for successful outsourcing is the possibility for the customer to ensure control of the software project during the entire software development

process. In this context, the requirements analysis, the system specification and the system design phases are the most critical phases with regard to outsourcing. Insight lost at these phases can normally only be recovered in further life-cycle phases with immense efforts. Therefore, these phases have been addressed by the *ESSI Project No. 24176 OutSource (Software Process Improvement Experiment Concerning Effective Outsourcing Mechanisms).* The results presented in this paper have been elaborated within the framework of this project.

Project Objectives

The OutSource project had the objective to improve the software development processes at the OutSource project partners with respect to outsourcing: Both project partners should be supplied with the best prerequisites to execute outsourcing projects just in time with higher quality and lower costs in reference to former software development projects.
In this context, special emphasis should be given to an improvement of the transparency and the controllability of the software development process from a management and a technical point of view.

The Project Partners and the Starting Scenario

The partners of the OutSource project are ATB Institute for Applied Systems Technology Bremen GmbH, a highly innovative and successful technology centre (prime contractor of the OutSource project), and BLG Automobile Logistics GmbH & Co., one of the largest port operating companies in Europe.
The idea for the OutSource project had the following origin: ATB had identified that a number of companies were reluctant to outsource software development because they were apprehensive of becoming dependent on the particular software suppliers. Therefore, ATB - in order to better serve customers and to increase its market share in the software business - wanted to establish a software development process which would remove such barriers at potential customers.
In contrast to ATB, BLG was faced with the following situation: In order to react to the increasing market requirements, BLG urgently needed future-oriented information systems for its operational business. Although BLG was supported by a central software department with respect to the development, installation and maintenance of operation-supporting software systems, BLG intended to emphasise outsourcing in the future in order to be more flexible and cost-efficient in software development and to reduce the time-to-market of new systems. However, BLG realised that for the effective management of outsourcing, its software engineering process had to be improved significantly.

The Baseline Project

The OutSource project has been connected to a baseline project in which ATB as a software supplier of BLG has developed an entire ‘Quality Information and Control System (QISS)’ concerning the loading and distribution of cars.

QISS is a typical business-oriented system which required efforts of more than 30 man months for realisation. QISS was urgently needed by BLG in order to maintain its competitive position as the leading port operating company in Europe for the import and export of cars. In the baseline project, BLG and ATB represented a typical "customer - software supplier" relation.

Key Aspects of the OutSource Experimentation

In order to achieve the objectives connected to outsourcing, the experimentation within the OutSource project focused on four key aspects which are closely connected and which are relevant for outsourcing altogether. These aspects are shown in figure CZW.1.

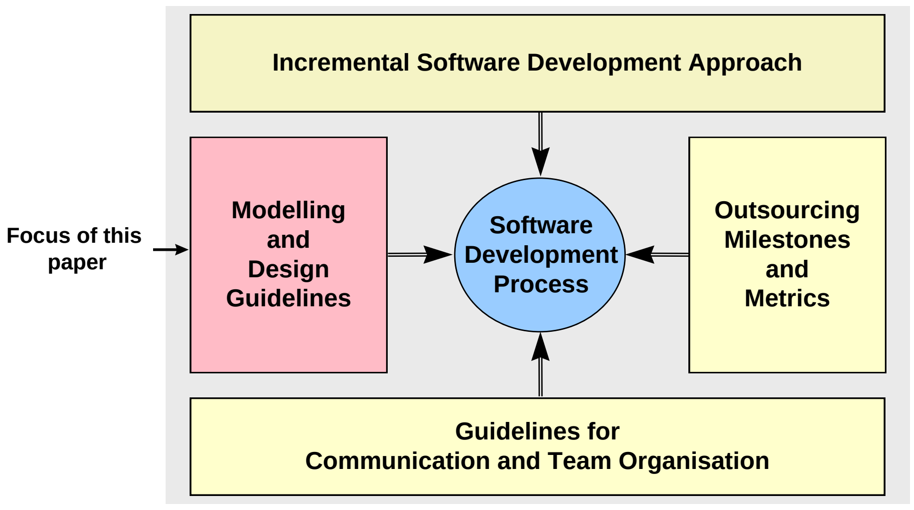

Incremental Software Development Approach
Focus of this paper
Modelling and Design Guidelines
Software Development Process
Outsourcing Milestones and Metrics
Guidelines for Communication and Team Organisation

Fig. CZW.1: Key aspects of the OutSource experimentation

The subsequent sections of this paper are - in addition to some remarks concerning the incremental software development approach - focusing on modelling guidelines which have been elaborated and tested in the framework of experimentation.

The Incremental Software Development Approach

The objective to provide sufficient project transparency and controllability to the customer can be successfully achieved by applying an incremental software development approach, the basic idea of which is to subdivide a software system to be developed into operational subsystems and subsystem expansions (see [7]). The first subsystem is an appropriate basic system of the final software system. By successive integration of the different subsystem expansions, the basic system evolves into the final software system (see figure CZW.2). The main benefits of this approach are:

- The development risk is minimised for the customer.
- The customer has good chances to get sound insight into the project progress.
- Feedback from real system operation is available at an early stage of the

project.

For the success of an incremental software development approach, it is essential that the subsystems and subsystem expansions are defined in an appropriate way. They should be tailored on the basis of

1. the business processes of the customer (in order to allow for a smooth software introduction) and
2. a global software specification (in order to secure that the different subsystem expansions will fit together).

Based upon the global system specification and the business processes of the customer (current processes and appropriately reengineered processes), a realisation stage plan should be elaborated which describes the relevant functionality for each realisation stage and its integration into the existing business processes. Each subsystem must have a high maturity level as a basis for its usability as an operational system. This requirement increases the amount of work that has to be performed for system integration and testing. Therefore, appropriate test procedures should be applied to reduce the work expenditure necessary to secure software quality.

It has to be emphasised that the structure of the subsystems and subsystem expansions is not necessarily correlated to the structure of the software components reflecting the internal software architecture.

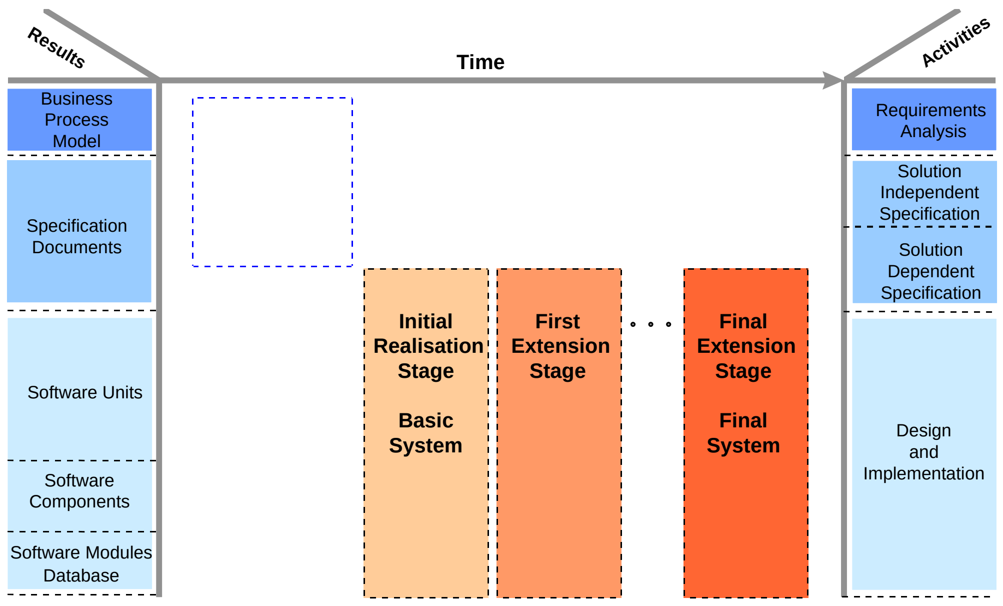


Fig. CZW.2: The incremental software development approach

An incremental software development approach is useful in all software development projects where there is a customer - supplier relationship. With respect to the special requirements of outsourcing projects, ATB has expanded the incremental software development approach by additional guidelines concerning technical and organisational aspects in order to further improve outsourcing project transparency and controllability on the part of the customer. The modelling guidelines presented below are one part of these additional guidelines. The resulting concept which was used and verified by ATB in field studies correlated to the above-mentioned baseline project is called ‘Incremental Software Development Concept for Outsourcing (ISDC)’. It has to be emphasised that the guidelines which have been developed and validated by experimentation are beyond the scope of definitions and guidelines which are made for example by the German V-Model (see [6] and [7]) or similar standards like for e.g. the DoD 2167A standard or the ESA software engineering standards (see [9]). The outsourcing guidelines are neither based upon nor in conflict with such standards.

Modelling Guidelines

A key essential in order to achieve a successful outsourcing process is to involve the user more intensively in the software development process throughout the whole development life cycle. The key challenge for this task is to find a common communication level between the users and the company management of the customer on the one hand, and the software development team on the other hand. To realise this communication, so called ‘models’ of information gathered and decisions made are required throughout the entire development process. A key criterion for an efficient communication process is a good quality of the created models. Experience gained not only in the OutSource project but also in several other projects has shown, however, that in modelling often one or more of the following typical problems occur:

- The modelling width is overly extensive in relation to the objectives of modelling, i.e. parts have been modelled which are not relevant with respect to the objectives of modelling.
- The modelling depth is overly extensive (too many details) in relation to the objectives of modelling.
- The models are poorly understandable and legible (because of missing guidelines with respect to naming, placing of symbols, general layout etc.).
- Different models are insufficiently comparable (because of insufficient guidelines for modelling).
- Within the models, consistency conditions are violated.
- The models contain useless redundancies.
- The number of hierarchical decomposition levels is too excessive.
- The models are poorly structured (different levels of abstraction are mixed).

The Methodology Reference Model

To overcome the above-mentioned problems, ATB has - as the first step of the experimentation - elaborated a so-called ‘Methodology Reference Model’ which describes different layers of topics for which modelling and documentational guidelines should be defined in order to reduce the corresponding degrees of freedom and by this to improve the communication process between the customer and the software supplier (see figure CZW.3). The Methodology Reference Model (the detailed structure of which is presented in [1]) can be used for any kind of modelling within the software development process.
Normally, if a special modelling method or modelling tool is applied, then only parts of the above-mentioned Methodology Reference Model are covered by the rules and conventions which are inherent in the method or tool. Especially the layers ‘Objectives of Modelling’, ‘Basic Modelling Principles’, ‘Attribute Generation’ and ‘Placing of Element Symbols’ are normally not

covered by common methods and tools. The other layers are often only partially covered or are subject to a configuration of the modelling tool. This implies that - according to the Methodology Reference Model - degrees of freedom remain which have to be covered by appropriate additional guidelines. Allowing the individual analyst to cover the remaining degrees of freedom without any additional guidelines, normally leads to the following problems:

– The modelling diagrams and documents concerning one project are difficult to read and understand, especially for the customer, because the remaining degrees of freedom are covered in a different way from diagram to diagram and from document to document. Usually, this effect increases significantly if more than one analyst is involved in modelling.
– It is even more difficult to compare the modelling diagrams and documents of different projects.

Therefore, a modelling method or tool to be applied has to be systematically analysed concerning which layers of the Methodology Reference Model are covered by this method or tool (a corresponding example is shown in figure CZW.3). The remaining degrees of freedom have to be covered by appropriate guidelines with respect to outsourcing. As a contribution to customer orientation of the software engineering process, special emphasis has to be given to the fact that such guidelines indeed do lead to models and diagrams which the customer can easily understand and verify.

Methodology Reference Model		**Additional Guidelines**	**Definitions by UML 1.2 and 'Rational Rose 98'**
Layer 1:	**Objectives of Modelling**	*A*	
Layer 2:	**Basic Modelling Principles**	*A*	
Layer 3:	**Element Types to be Used**	*A*	T
Layer 4:	**Views to be Created**	*A*	T
Layer 5:	**Modelling Syntax**		T
Layer 6:	**Attribute Generation**	*A*	T
Layer 7:	**Representation of Elements (Element Symbols)**	*A*	T
Layer 8:	**Placing of Element Symbols**	*A*	
Layer 9:	**Structures and Layout of Reports**	*A*	T

Fig. CZW.3: The different layers of the Methodology Reference Model

Based on the above-mentioned Methodology Reference Model and in reference to the outsourcing requirements, for the following types of models appropriate guidelines have been applied and tested in the scope of experimentation:

1. Business process models based upon ATB's 'Task Analysis Methodology (TAM)' (created with 'ABC FlowCharter' in combination with 'MS Access')
2. ARIS-type business process models (created with the ARIS Toolset)
3. Functional models (created with the 'Select SSADM' CASE tool, see [10])

4. Entity relationship models (created with 'Select SSADM')
5. Use case models (based upon UML and 'Rational Rose 98')
6. Class diagrams (based upon UML and 'Rational Rose 98')
7. Sequence diagrams (based upon UML and 'Rational Rose 98')

The model types 1 and 2 have been applied in the scope of experimenting with different approaches for business process modelling as a pre-step of software development (see [1], [2], [3] and [15]). The model types 3 and 4 have been applied in the scope of experimentation with base technologies which belong to a more classical approach for system specification, whereas the model/diagram types 5, 6 and 7 are part of the relatively new Unified Modeling Language Version 1.2 (UML 1.2, see [5], [8], [13] and [14]) the importance of which, however, with respect to outsourcing projects will significantly increase in future. In the following sections, some examples of the results achieved are given.

Functional Model

In the baseline project, the project partners elaborated - based upon the business processes of the customer - a solution-independent system specification which consists of a functional model and an entity relationship data model.
As a part of the OutSource project, different representation approaches for the functional model have been applied and tested. Included in the experimentation were data flow diagrams, function tree diagrams and function/entity matrices showing the correlation between functions and data entities.
Experimentation disclosed that data flow diagrams require a lot of layout work in order to make these diagrams easily readable. The benefits yielded by data flow diagrams proved inadequate when compared to the efforts required for the layout work concerning these diagrams. As a consequence, data flow diagrams were elaborated only for presentations of the context situation, of the 1st functional decomposition level and of some branches of the second functional decomposition level. For the presentation of the entire functionality of the software system, function tree diagrams proved to be very useful. For the specification of the correlation between functions and data entities, function entity matrices were applied successfully. The individual functions of the system were described in a textual manner without any additional guidelines to describe the functional flow of events.

Use Case Model

The more classical approach of a functional model was compared to a use case model approach (see [11] and [12]) which is compliant with the future-oriented Unified Modeling Language (UML). Based upon the business processes of the customer, in the scope of the OutSource experimentation parts of the system functionality were modelled on the basis of use cases with the aid of the 'Rational Rose 98' CASE tool. This use case model is an alternative part of the solution-independent system specification.
The decision to use the 'Rational Rose 98' CASE tool had on the one hand company-strategic reasons but was on the other hand also based on the

results of a comparison of 'Rational Rose 98' to the 'SELECT Enterprise Modeler' CASE tool with respect to use case modelling. The comparison showed that 'Rational Rose 98' has a wider spectrum of functionalities for example concerning the possibility to define associations between actors or the possibility to refine every model element with different types of diagrams. During experimentation, due to the existence of a higher number of different use cases, the necessity of an appropriate grouping of use cases was identified in order to yield better understandability and legibility of the use case diagrams. The grouping was carried out analogous to a classical functional hierarchy.

The first approach for the grouping of use cases was to use the UML construct of packages. In this context, the intention was to show all associations between actors and use cases not only on the use case level but also on the package level. This, however, is not possible because associations between actors and packages are not allowed within UML 1.2 (see [13]) and 'Rational Rose 98'.

As an alternative approach, use cases with special stereotypes were used for the logical grouping of use cases. For all stereotypes of use cases, the corresponding actors and their associations (interactions) with the use cases were included in the diagrams. After several refinements, the stereotypes 'system', 'process', 'operation' and 'sub-operation' were defined: The 'system' stereotype was used in the top-level (level 0) use case diagram. This diagram is a context diagram which contains on the one hand a use case representing the software system with all its use cases, and on the other hand all the different actors (roles and external systems) including their interactions with the system. The 'process' stereotype was used for all use cases below the context level which represent a logical grouping of other use cases. For every use case with the 'process' stereotype, a separate use case diagram was created. The 'operation' stereotype was used for all 'original' use cases, i.e. use cases which do not represent a logical grouping of other use cases. By means of the 'sub-operation' stereotype, all use cases were indicated which appeared as a consequence of the 'uses' and the 'extends' associations and which do not represent independent and complete operations.

In correlation with the above-mentioned Methodology Reference Model, guidelines were elaborated which include - among other things - the naming of use cases, the structure of use case descriptions, the placing of element symbols in use case diagrams and the structure and layout of use case model reports.

Experimentation has shown that the semantic approach for the definition and identification of use cases is very useful in reference to the outsourcing requirements. In classical functional modelling, it is often not obvious which sets of functionality should be defined as functions and it is often difficult to identify whether important functionality which should belong to a specific function has not been included. These problems have been solved quite practically for use cases by the inventor of use cases, i.e. by I. Jacobson (see [11] and [12]). Experimentation, however, showed that the use case diagrams are not as easily legible and understandable, as compared for example to function tree diagrams. Therefore, an approach for future activities could be the use, for example, of classical function tree diagrams in which, however, every elementary function has the identity of a use case.

Figure CZW.4 shows an example of a level 2 use case diagram which has been created as a result of experimentation.

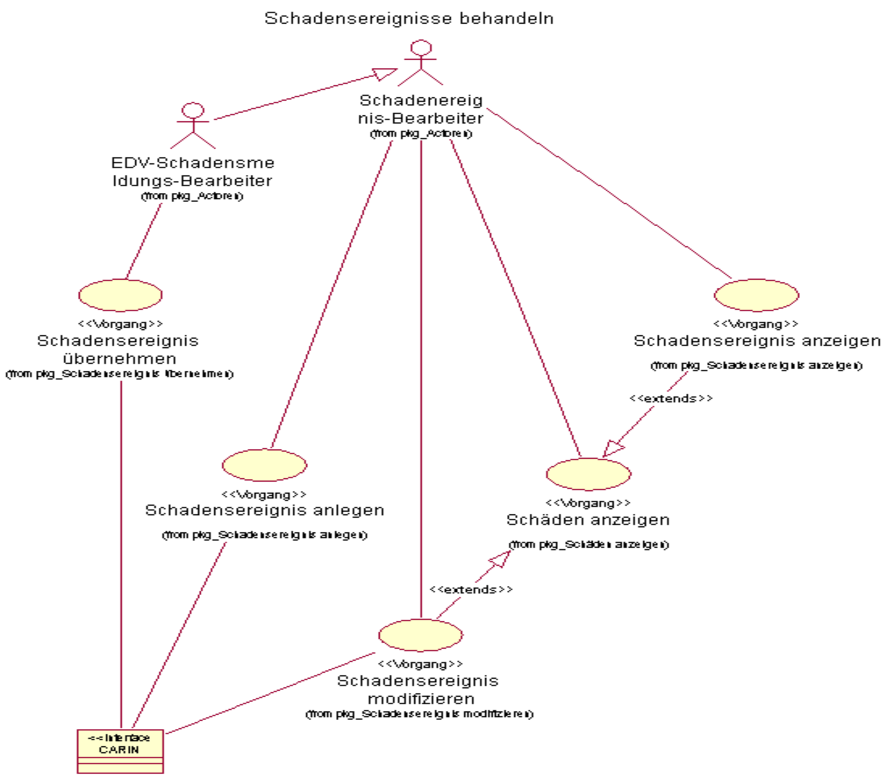

Schadensereignisse behandeln
Schadenereig nis-Bearbeiter
(from pkg_Actoren)
EDV-Schadensme ldungs-Bearbeiter
(from pkg_Actoren)
<<Vorgang>>
Schadensereignis übernehmen
(from pkg_Schadensereignis übernehmen)
<<Vorgang>>
Schadensereignis anzeigen
(from pkg_Schadensereignis anzeigen)
<<extends>>
<<Vorgang>>
Schadensereignis anlegen
(from pkg_Schadensereignis anlegen)
<<Vorgang>>
Schäden anzeigen
(from pkg_Schäden anzeigen)
<<extends>>
<<Vorgang>>
Schadensereignis modifizieren
(from pkg_Schadensereignis modifizieren)
<<Interface
CARIN

Fig. CZW.4: Example of a level 2 use case diagram

Class Diagrams

Based on the use case model and also as a part of the solution-independent system specification, hierarchically structured class diagrams were created. In this context, it should be emphasised that - according to the well-proven principle of separation between solution independent and solution dependent specification - these class diagrams (called 'specification class diagrams') should be structured with respect to customer requirements and not with respect to a desired software architecture. The different classes occurring during specification were structured into different packages in order to facilitate an easier communication with the customer.

It has proved to be recommendable that the specification class diagrams should include not more than two or three hierarchical decomposition levels: The top level (level 1) specification class diagram contains only the different packages and their relationships (see figure CZW.5). On the second hierarchical level, for every package a separate specification class diagram is created which contains on the one hand all classes of the corresponding package including all the class attributes and operations, and on the other hand those classes of other packages to which direct relationships exist. Classes belonging to other packages are represented without attributes and operations. In cases of a bigger number of classes, it may be necessary to present in some branches of the second hierarchical level again packages

instead of classes. In this case, for those branches a third hierarchical level is required.
In addition to these diagrams - if desired - class diagrams can be created which contain all classes with their interactions. Dependent on project needs, these diagrams can include only class names but can also include class attributes and/or class operations, if desired.
In correlation to the Methodology Reference Model, guidelines were elaborated and tested which include - among other things - the naming of classes, attributes, operations and relationships and the placing of element symbols in the class diagrams.

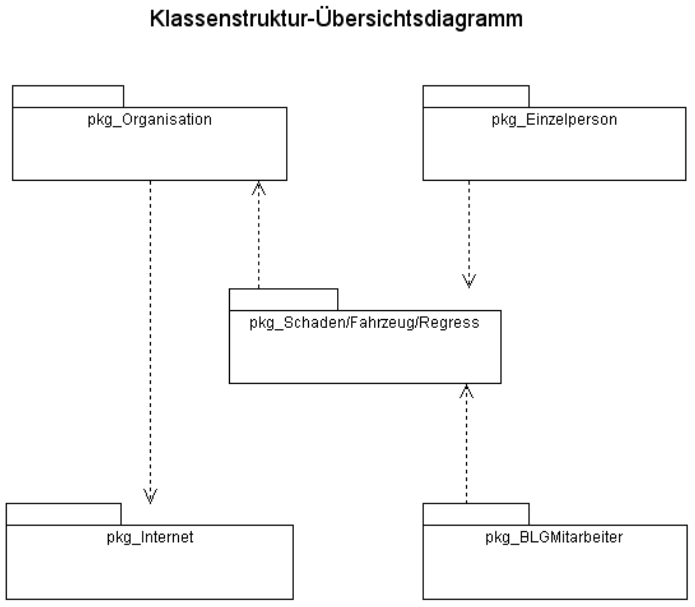

Klassenstruktur-Übersichtsdiagramm
pkg_Organisation
pkg_Einzelperson
pkg_Schaden/Fahrzeug/Regress
pkg_Internet
pkg_BLGMitarbeiter

Fig. CZW.5: Example of a top level (level 1) specification class diagram

When the specification class diagrams have reached a certain level of maturity, the corresponding class model has to be re-structured under aspects of optimal software architecture, optimal software design and software re-usability. The re-structuring results in a set of 'design class diagrams' for which - in reference to the outsourcing requirements - also appropriate guidelines were elaborated and tested.

Sequence Diagrams

In several software development projects, ATB has applied a classical modelling approach including a functional model and an entity relationship data model, but not including a special dynamic model. This modelling approach was also applied in the baseline project since dynamic aspects were relatively simple to handle. Due to this fact, in the OutSource experimentation only for a few selected scenarios dynamic modelling has been carried out using sequence diagrams.

Measurement of Results

In the scope of the OutSource project, various parameters were measured in order to assess the different improvement actions. Measurement parameters were communication and team organisation parameters (e.g. the duration and frequency of meetings), change request parameters (e.g. the number of change requests and the efforts per change request), general effort parameters (e.g. efforts for integration, testing and installation), milestone parameters (e.g. time intervals between milestones), legibility parameters (legibility of the different diagram types), quality parameters (e.g. the quality of the software system w. r. t. functionality and usability and the number of errors grouped into different error classes) and business parameters (e.g. the increase of ATB's annual turnover in the software business). The measurement results indicate that the objectives of the OutSource project have been fully achieved.

Lessons Learnt

In the following, an extraction of the most important lessons learnt is given which are derived from the OutSource experimentation. The lessons 1 - 3 refer to the incremental software development approach, whereas the lessons 4 - 7 refer to modelling aspects.

1. By the use of an incremental software development approach, customer satisfaction can be increased significantly. An incremental software development approach, however, requires additional organisational, integration, testing and communication efforts compared to a more classical software development approach.
2. During the course of the incremental software development approach, temporarily unanticipated efforts occurred which were caused by insufficiencies in the applied test procedures, resulting in unexpected incompatibilities and inadequate reliability levels of some operational extension stages (subsystems).
3. The number of change requests increased significantly compared to former baseline projects because the customer was encouraged to articulate change requests in order to increase software quality.
4. Software tools for modelling leave quite a lot of degrees of freedom which - if they are not reduced by additional guidelines - lead to models of bad quality, among other things, with respect to the quality characteristics 'understandability' and 'legibility'. Such models complicate the communication process between the customer and the software supplier in a significant manner. In order to improve the communication process, additional guidelines for modelling and documentation have to be defined and applied which reduce the degrees of freedom remaining by the applied modelling methods and software tools. With the aid of efficient modelling and documentation guidelines, a consistent and by all partners easily readable documentation can be achieved. As a consequence, the efficiency of meetings and interviews and by this the quality of feedback information from the customer can be significantly increased.
5. Experimentation within the OutSource project confirmed previous ATB experiences that a high abstraction level of applied methods for the acquisition of user requirements can lead to significant

misunderstandings between the project partners, process inefficiencies and expensive failures of the software. In order to avoid such problems, for the acquisition of user requirements, methods should be applied which have a low level of abstraction. An example of such a method is the 'Task Analysis Methodology (TAM)' which was developed by ATB and further refined in the scope of the OutSource experimentation with respect to outsourcing requirements (see [2]).

6. Redundancy in modelling reports can bear significant advantages compared to a non-redundant presentation. Especially for reports which had to be intensively discussed with the customer, structures were chosen in which for every diagram the textual descriptions of all occurring modelling elements had been grouped together below the diagram. Due to the fact that the same modelling elements occur in several diagrams, such a structure leads to redundancy in the presentation. The advantage, however, is that the individual diagrams are more easily readable because all corresponding textual descriptions are grouped together, i.e. the customer does not have to search for textual descriptions in different chapters of the report. It has to be emphasised, however, that all textual descriptions should be kept in a tool database in a non-redundant manner because otherwise the effort for changing textual descriptions would be unacceptable.
7. For the specification of software functionality, the semantic approach for the definition and identification of use cases proved useful. Experimentation, however, showed that the use case diagrams are not as easily legible and understandable for the customer, as compared for example to function tree diagrams.

Conclusions and Future Plans

As a result of the experience gained in experimentation within the OutSource project, the project partners ATB and BLG have improved the quality of their software engineering processes significantly with respect to the outsourcing problematic. Both project partners intend to apply the results in future software development projects. The results obtained in the OutSource project have a high potential for external replication because they are beneficial not only for software suppliers but also for customers with the intention to practise outsourcing.

Based on the results which have been achieved within the OutSource project, ATB and BLG have set up company-internal implementation plans concerning upcoming future activities for further improvements of their software engineering processes with respect to outsourcing requirements: ATB will continue the improvement of its software development process, especially with respect to the reduction of unforeseen efforts connected to an incremental software development approach and the elaboration of appropriate outsourcing guidelines for additional UML diagram types (for e.g. activity diagrams and component diagrams). Beyond that, ATB will use the OutSource results as marketing arguments in order to further increase ATB's annual turnover in the software business. BLG will - based upon the results of the OutSource project - further improve its internal business processes especially with respect to the cost-efficient management of outsourcing projects. Beyond that, BLG will define types of projects and development tasks for which outsourcing will be intensified in future.

References

[1] ATB Institute for Applied Systems Technology Bremen GmbH: *ESSI Project No. 24176 OutSource: Software Process Improvement Experiment Concerning Effective Outsourcing Mechanisms*, Final Report, Bremen (Germany), December 1998

[2] ATB Institute for Applied Systems Technology Bremen GmbH: *Modellierung von Geschäftsprozessen mit TAM: Richtlinien und Festlegungen*, Document-Id.: SW-QM-AA-ATB-CZW-026/02, Version 1.1, Bremen (Germany), September 1998

[3] ATB Institute for Applied Systems Technology Bremen GmbH: *Modellierung von Geschäftsprozessen mit ARIS: Richtlinien und Festlegungen*, Document-Id.: SW-QM-AA-ATB-CZW-022/03, Version 1.2, Bremen (Germany), October 1998

[4] ATB Institute for Applied Systems Technology Bremen GmbH: *Modellierung mit UML und Rational Rose: Richtlinien und Festlegungen* Document-Id.: SW-QM-AA-ATB-CZW-023/02, Version 2.0, Bremen (Germany), May 1998

[5] Burkhardt R.: *UML - Unified Modeling Language, Objektorientierte Modellierung für die Praxis*, Addison-Wesley, 1997, ISBN 3-8273-1226-4

[6] Der Bundesminister der Verteidigung (BMVg): *Entwicklungsstandard für IT-Systeme des Bundes: Vorgehensmodell (V-Modell) Version '97*, Interaktive CD-ROM Version '97, October 1997

[7] Dröschel W., Heuser W., Midderhoff R. (Editors): *Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97*, R. Oldenbourg Verlag München Wien 1998, ISBN 3-486-24276-8

[8] Eriksson H.-E., Penker M.: *UML Toolkit*, John Wiley & Sons 1998, ISBN 0-471-19161-2

[9] European Space Agency: *ESA Software Engineering Standards Issue 2 (ESA PSS-05-0 Issue 2)*, Noordwijk (The Netherlands), February 1991

[10] Goodland M., Slater C.: *SSADM Version 4, A Practical Approach*, McGraw-Hill Book Company Europe, 1995, ISBN 0-07-709073-x

[11] Jacobson I., Christerson M., Jonsson P., Övergaard G.: *Object-Oriented Software Engineering, A Use Case Driven Approach*, Addison-Wesley Publishing Company; ACM Press, 1992, ISBN 0-201-54435-0

[12] Jacobson I., Ericsson M., Jacobson A.: *The Object Advantage - Business Process Reengineering with Object Technology*, Addison-Wesley Publishing Company; ACM Press, 1995, ISBN 0-201-42289-1

[13] Object Management Group, Inc.: *OMG Unified Modeling Language Specification, Version 1.2,* July 1998

[14] Quatrani T.: *Visual Modeling with Rational Rose and UML,* Addison-Wesley, 1998, ISBN 0-201-31016-3

[15] Scheer A.-W.: *Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse,* Springer-Verlag, 1994, ISBN 3-540-58203-7

[16] Zwanzig Ch., Schulz H.S.: *The Incremental Software Development Concept: A Key Aspect for Successful Outsourcing,* in: Proceedings of the European Software Day (EUROMICRO '98 Conference), pp. 71-88, Västerås (Sweden), August 27, 1998

Appendix

ATB Institute for Applied Systems Technology Bremen GmbH

ATB Institute for Applied Systems Technology Bremen GmbH, located in Bremen (Germany), was founded in 1991 and has the status of a non-profit organisation. The shareholders of ATB are the State of Bremen and a group of important industrial companies. Among others, ATB shareholders are Daimler-Benz AG, STN ATLAS Elektronik GmbH, Bremer Lagerhaus-Gesellschaft AG and also a number of medium-sized enterprises which are located in the region of Bremen. ATB has established a research team of 17 scientists which is supported by about 20 employees working on a temporary or part-time basis. ATB employees have expertise and sound experience in manufacturing, logistics, control theory and software engineering. ATB operates on the basis of a quality management system which is fully compliant with ISO 9001. The main strategic business areas of ATB are software systems technology, systems analysis & design and quality management.

One basic objective of ATB is to apply system technology methods and tools in order to improve the quality and efficiency of processes in various application areas. Special emphasis is given to the improvement of software engineering processes. The general concept of ATB is to establish in close co-operation with its industrial partners well-balanced application-oriented research activities, which should result both in solving the specific problems of the partners, and in contributing to the further development of system technology sciences. About one third of ATB's annual turnover is realised in the business area of software systems technology. Within this domain, ATB has gained a comprehensive overview concerning the large variety of software best practice methods and tools available on the market. ATB is applying and evaluating these advanced software engineering technologies in the scope of developing application specific software systems for industrial customers.

Curriculum Vitae of the Author

Name:	Christian Zwanzig
Age:	42
Education/Degree:	Dr.-Ing.
Experience:	1981 - 1989 Research assistant at the Institute for Control Theory and System Dynamics of the Technical University of Berlin. Research in the field of non-linear and adaptive control theory; lecturing on digital signal processing. Co-operation in an industrial project to develop a lab prototype. Co-author of a textbook on non-linear and adaptive control theory. 1989 - 1995 Project engineer at STN ATLAS Elektronik GmbH in Bremen. Section leader for software engineering concerning hydrographic data acquisition systems and marine environmental monitoring systems. Marketing activities in the area of marine environmental monitoring systems. Strong involvement in the establishment of a company-wide software engineering norm. Since 01.09.1995

Senior researcher at ATB Institute for Applied Systems Technology Bremen GmbH, responsible for ATB's business area of software systems technology. Involved in all software process improvement activities at ATB. Leader of several industrial software development and business process reengineering projects. Execution of training seminars on various software topics.

CCM - A fundamental Process for Improving Quality

Clemens Gasser
Joanneum Research, Graz, Austria
Edwin Deutschl
Joanneum Research, Graz, Austria.

1. Introduction

This paper describes the practical work done and the experience made during the introduction of Change and Configuration Management (CCM) – a key process in software developing - in a small department of a RESEARCH company, which has a certified ISO-9001 system in whole, but no formal defined or practically used software developing process in detail.

The goals of the Process Improvement Experiment (PIE) were to shorten the software developing cycle, to increase the reuse of source code by introducing a well defined Software Developing Process (SDP) and a CCM. Faster maintenance by better reproducibility and readability of code and by a better documentation were further goals of the project.

Up to now, the SDP and the CCM-plan were defined and introduced to the baseline project DIBIT, a photogrammetic measuring system for the documentation of tunnel advances and underground constructions. This project is in an intensive maintenance phase, with steady new releases and variants for different customers and therefore best suited for an experiment with CCM. In the moment the measurement phase of the PIE is in progress.

First results and key lessons learned are, that the introduction of the CCM-tool was more time consuming than planned and that it is very hard to find expressive metrics.

The next proposed actions are the consolidation of the introduced processes and better documentation of them. But the focus will lie on the metrics, which data gathering process is in full progress. International and internal dissemination are further actions to propose.

Background Information

The *Industrial Image Analysis* department of the *Institute for Digital Image Processing* at JOANNEUM RESEARCH develops customer applied systems for industrial fields concerning image processing technologies [7]. The list of customers reaches from steel-, forest-, pharmaceutical- and automotive industry to tunnel construction (see also Appendix II: About JOANNEUM RESEARCH).

Most of the systems are built as prototypes, which are adapted to the customers and their needs. In recent years systems were placed on the market, that were produced in a higher number of units. This required, in addition to normal maintenance for a prototype, further development and adoption to customer specific variants.

Additionally, most of the systems require a software development that depends extremely on hardware restrictions (Intelligent cameras, framegrabber, sensor interfaces, SPS process control, heavy duty environments, etc). Several systems have to fulfill real-time requirements (e.g. optical inspection systems in production lines). This special requirements need a well defined and quality oriented software engineering.

The experiment should help to get closer to a high level software engineering at the department by introducing the key process **Configuration Management** as an important supporting activity in software development.

2. Starting scenario

In 1995 JOANNEUM RESEARCH was certified to the ISO 9001 standards. Thereby the attention was directed at administrative processes. Within JOANNEUM RESEARCH there are only a few institutes concerned with major software development. Because of this fact, certification of the software development process was not accomplished.

The state of software development within the department *Industrial Image Analysis* at the beginning of the experiment depends strongly on the initiative of the individual project leaders. No common guidelines or defined processes existed for the entire department. The major problems, like exploding budgets or overrun milestones, were partially lead back to inefficient software development.

An overview of the strengths and weaknesses in software development at the starting point of the IECS was received by a SynQuest-Assessment, which is based on the Bootstrap method [13]. These assessments are performed at the beginning and at the end of the experiment. They are used to establish the impacts of the newly introduced processes in the attitude and behaviour of the employees. The guided SynQuest assessment showed the following results concerning strengths and weaknesses of the organisational unit.

The strengths:	The weaknesses:
Organization and project management.	Configuration- and change-management.
Well pronounced responsibilities.	Testing.
Comparably high degree of	Metrics and inspections.

training.	

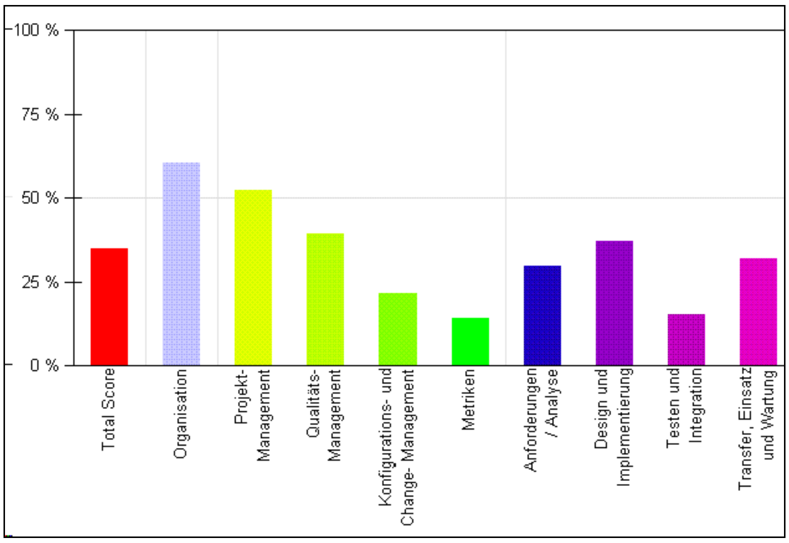


Fig. CGAS. 1: Results of the SynQuest assessment – Process fields

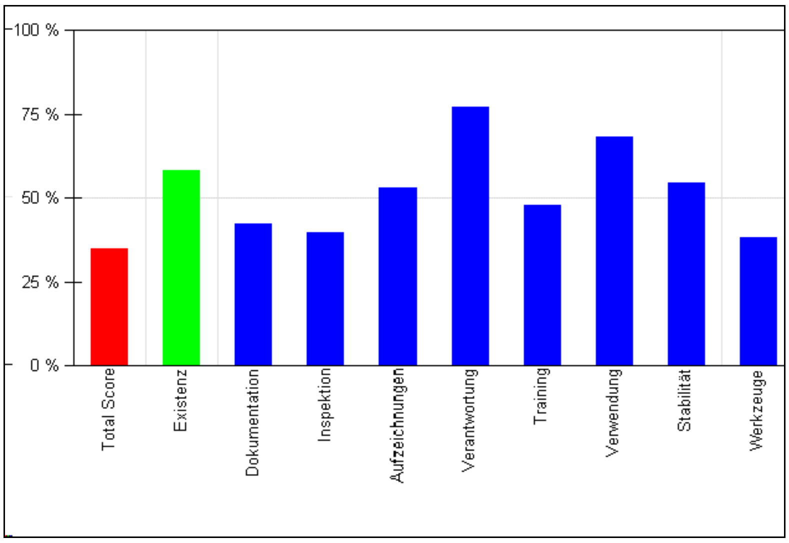


Fig. CGAS. 2: Results of the SynQuest assessment – Process attributes

Fig. CGAS. 1 and Fig. CGAS. 2 show the average results of four project groups, each containing three or four employees.

Apart from common organisational aspects the maturity of the software developing process could be judged as ad-hoc and therefore comparable to a CMM level 1.

Split up into the following different points of view the analysis of the strengths and weaknesses brought up:

Technical

The employees used regularly software development tools, like C/C++ compilers together with integrated developing workspaces (e.g. Microsoft Developer Studio, Microware Os9 – Cross Compiler). Software developing was focused on the source-code and low-level debugging, rather than on supporting activities like configuration management or documentation. Only a central administrated GroupWare tool (DIGITAL Linkworks) supported corporate working, especially in documentation and project management. The absence of a shared source code repository lead to different variants of equal software components also among individual employees. No co-ordinated software reuse existed. The use of different hardware-platforms also complicated the software developing.

Business

Most of the systems are developed completely new and are therefore very cost intensive. On the other hand such expensive systems can hardly be sold to customer. So there had to be made some retrenches, which often concerned the part of budget for software development. Because customer can easier grasp the amount for hardware than for software.

Organisational

Because of the small sized organisation, lean organisational structures are used. The head of department is supported by a number of project managers, who themselves are involved in different projects and who lead a small number of collaborator. The typical size of project teams is about three employees. How the SynQuest assessment confirmed, organisation was on a high level. The reasons therefore may be find in the ISO-9001 certification as mentioned above. This high level of organisation concerned more to common aspects, than to software development. The guidelines ISO 9000-3 were nearly not established.

Cultural

Most of the employees of the department are graduated or technical engineers and are from their nature very interested in new technologies and new working practices. It could be observed, that older engineers, who have consolidated working practices are more sceptical against new techniques than younger one.

Skills

The base skills of the employees can be summarised as technical experts in the field of electrical engineering, algorithms and mathematics. The skills in software development evolved in practical work or - in other words - in learning by doing. Most of the baseline project team members master the programming language C, but more from an algorithmically point of view than from an informational. None of the employees had advanced experiences in software engineering or software quality. The average time of experience in programming of the baseline project team was about seven years. One team member had also skills in the language C++ for implementing user interfaces.

3. Expected outcomes

Most of this expected outcomes concern to the maintainability quality characteristic.

Shorter software development cycles to reduce personal and resource costs (obviously a very common objective).

- Higher quotas of software reuse to reduce costs for development and to increase the software reliability .
- Increase of reproducibility and readability of code for easier and faster maintenance.
- More efficient configuration and change management for higher maintainability and portability.
- Introduction of problem tracking mechanisms to guarantee consistent and permanent correction of errors and increasing usability.
- Better documentation for easier maintenance.

4. The plans and their implementation

This section describes in detail the working plans and their implementation in the ESSI-PIE.

Organisation

The IECS project is organised in a structure as presented in Fig. CGAS. 3. The head of the department reviews and controls the project from a business point of view. The main work of the IECS-project is performed by the project manager of IECS. On his side, a so called Software Quality Group (SQG, comparable with SEPG in [8]) was installed.

The IECS project manager co-operates intensively with the project manager of the baseline project as well with the baseline project team in the sense of the objectives of the experiment.

It is worth mentioning, that a lean organisational structure has been chosen in respect to the limited personal resources of our department as an SME.

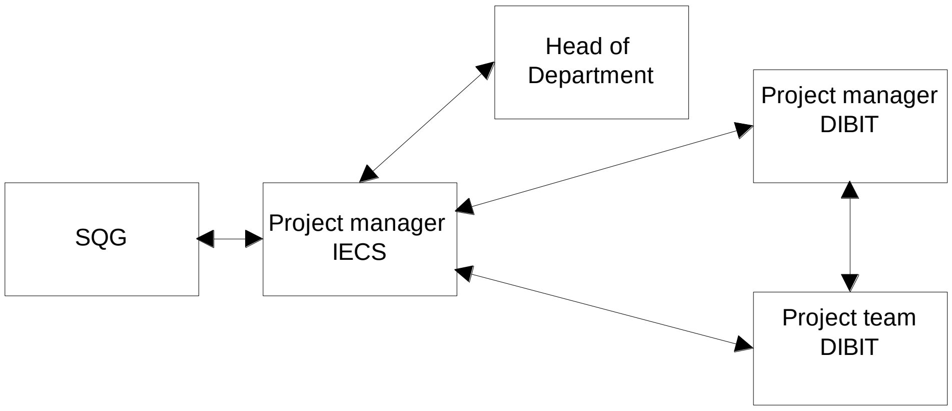


Fig. CGAS. 3: Organisational structure

Technical environment

Main changes in the technical environment were necessary for the installation of the CCM – Tool. A Unix File-Server was installed on a SUN-workstation under the Solaris Operating system. It serves as repository for the CCM – Tool as well as an Internet Server (Netscape Enterprise) for the new introduced Intranet DIBIS (DIB – Information System). This Intranet supports the transmission of the new methods and the access to guidelines, templates and forms to all employees involved in the experiment.

MKS Source Integrity [11] was selected as CCM – Tool. It was evaluated as best fitting to our criterions for a configuration management tool based on the CCM - plan. The following important requirements were identified to be served by the tool:

- Platform independence (Solaris Sun and Windows/PC).
- High usability (short time for learning to use the tool appropriately).
- Good ratio of price to performance.
- Common CCM - functionality.
- Guaranty of a good technical support.

Other evaluated CCM - Tools are listed in Table 1.

For metric purposes the tool Provista from 3Soft was installed. As metric databases MS-Access and MS-Excel are in use. For documentation of both customer and internal requests a MS-Access database was designed and installed.

Tool	Firm
PVCS	Intersolv, USA
Source Integrity	MKS, BRD
Lifespan	BAeSema, UK
Continuus/CM	Continuus Software GmbH., BRD
ClearCase/ClearCase Attache	Atria, USA

Voodoo	Uni Software Plus, Austria
Versionmaster	Soft Systems, Canada
SourceSafe	Microsoft, USA
RCS f. UNIX	PD Software

Table 1: Evaluated CCM - Tools

Training

Selected topics of the Definition of the Software Developing Process and the CCM – plan were presented in individual sessions at the department to all employees. An external expert in software engineering from the University of Dundee held a training lesson about ”Software quality and the Software Live Cycle”.

Most of the training for an efficient use of the CCM Tools was done by training on the job of the concerned staff members. A preceding self-training, by studying the manuals, making pilot-experiments with example-projects and contacting the software supplier in difficult questions, puts one person into ability to train the others.

Self-training can be done by every employee by studying the guidelines and documented processes either by the GroupWare tool LinkWorks or by the Intranet DIBIS. Two people were also trained by an external consultant into an overview in the methods and practises of CCM.

External consultants

In the starting phase of the experiment, an external consultant was engaged, to train some employees on CCM. He also formulated the draft version of the CCM-plan.

An other external consultant was engaged for supplying the guided SynQuest Assessment. He guided the whole assessment and worked out the documentation of the results. In a separate session he presented the results to the employees of the department.

Phases of the experiment

The IECS project splits up into the following planned main phases:

Phases	Deliverables
Project management and co-ordination Contacts to EC, periodic reporting, planning, communication	Mid Term report, Final report, Cost statement
Basic Assessment Holding of a guided SynQuest assessment	Basic Assessment report
SDP definition Definition of the phases of the Software developing process	PES
CCM process definition Definition of the CCM-Plan and requirements for a CCM-Tool	CCM-Plan (draft)
CCM - Tool selection, installation, training	CCM-Evaluation report
Experiments, improvements and surveillance Work with the baseline project, definition of measurements	Measurement reports
Final assessment (Not actually performed) Holding of a guided SynQuest assessment and comparison with the initial assessment	Final Assessment report
Reporting and dissemination (Not fully performed) Writing of final reports and papers and preparation for dissemination events.	CCM-Plan (final), Final report

The work, which is actually performed, will be described in the following section.

Project planning: The first phase of the IECS was characterised by project planning and settling into the topics of the experiment. That means, that adequate personal resources were fixed and a detailed work-plan was performed. The installed project manager was not involved in acquiring the PIE, so there was an additional amount to get into the project topics. Special literature had to be obtained and had to be studied.

Basic Assessment: In parallel the initial basic SynQuest Assessment was organised. The assessment was guided by a professional consultant of HM&S, the company which developed the SynQuest tool. The experience and results of keeping a lot of guided assessments in other companies provides a possibility to compare the own maturity-level in software developing with that of other organisations. The consultant showed this comparison at the presentation of the assessment results. The average results lies clearly over the average result of all assessed organisations with the exceptions CCM, metrics and testing.

SDP definition: Also in parallel the SDP was worked out and documented in the PES. For practicability it was attempted to keep it as compact and clear as possible. The formulation of the PES was done with respect to practices and standards already used by the engineers. The resulting subdivision of phases (see Table 2) is related with the classical Waterfall Lifecycle model. It is still used by a wide area of standards and guidelines (see ISO 9000-3 [9], TickIt [5], CMM [8] and the summary of [14]). As part of the SDP Coding conventions were worked out. They define a uniform appearance of the source code and are therefore an essential method to increase the software readability.

Phase	Purpose	Outputs
User Requirements	Problem definition	"User Requirements Document", test plans, Project Management Plan
SW Speci-fication	Problem analysis, Logical model	"SW Specification Document", test plans
Design	System design, Physical model	"Architectural Design Document"
Production	Software implementation, tests	"Detailed Design Document", Source code, User Manuals
Transfer	Installation and acceptance	"Software Transfer Document", acceptance protocols
Maintenance	Maintenance, improvements	Maintenance protocols, SW-Releases

Table 2: The phases of the software developing process (SDP)

CCM-Process definition: The phase of defining the CCM-Process was supported by an external consultant. In initial sessions the CCM-requirements depending on the SDP were defined. Individual CCM – activities were discussed and documented in a CCM – plan (see Table 3), to which various standards gave helpful hints. Again the guidelines of ISO 9000-3 [9], TickIt [5] and CMM [8] were taken into account and used as templates.

CCM – plan topics	Description
Configuration management	Description of the CCM organisation, CCM responsibilities, relationship to the SDP.
Configuration identification	Specification identification, Identification of the baseline
Change control procedures	How to Check-In, Check-Out, make checkpoints.
Build management.	How to build new releases.
Change Requests.	How to handle requests, bugs, modifications.
CCM - Tools, Techniques, Repository.	Specification and description of the technical environment.
CCM - Reporting.	How to trace changes, and make report about them.

Table 3: Overview of the CCM-plan

CCM - Tool selection, installation, training: The next phase was the evaluation and selection of an adequate CCM – tool. First a list of potential tools and their suppliers were prepared. The evaluation was based on the inspection of demo versions, documentation material and information received from the tool suppliers. The best rating on this evaluation was accomplished by the tool packet Source Integrity by MKS [11]. Later good references for CCM – tools were found in the Internet, which get a more and more valuable information pool (see [15] as a very extensive storehouse for configuration management). The installation and initial administration of the CCM - tool spent more time than estimated. A lot of small problems (technical but also in comprehension) must be taken into account at introducing a totally new tool.

Experiments, improvements and surveillance: The following phase concerned the work with the baseline project. The originally provided baseline project had to be dropped, because it fells in a status, where no further development. Therefore it was necessary to take an other baseline project. A project for an optical 3D-surface reconstructing tool was selected. This project is called DIBIT (Digital Image Observation System for Tunnelling). The DIBIT tunnel scanner (see the system in practical use in Fig. CGAS. 4) is a photogrammetic measuring system for the documentation of tunnel advances and underground constructions.

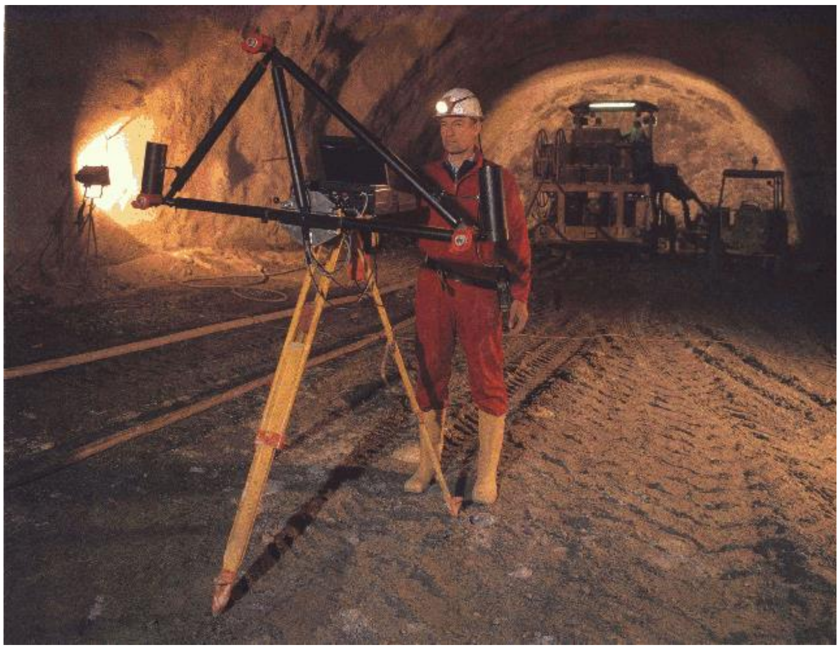


Fig. CGAS. 4: The DIBIT system in practical use.

It was developed during the last four years at our institute. At the moment there are three delivered systems in practical use. Each system has varying configurations for the different tunnels and requirements of the customers. Furthermore there must be a handling of error requests and new requirements, which come in permanently. The main problems to solve concern to configuration management.

After the approval for the exchange of the baseline projects the initial work for the new baseline project had to be performed. The theoretically outworked **measurement program** was adapted to practical use for the experiment with the new baseline project. For evaluation of appropriate metrics the GQM (Goal - Question - Measurement) method [14], [12] was used. Four classes of metrics together with the results of the SynQuest assessment form the essential points of the metric program (see Table 4):

QO.1 defines the transparency in amount of time for different tasks, performed either for IECS and the baseline project. The spent time is recorded in Excel forms (see Fig. CGAS. 5) by all involved employees. The forms are analysed each month and stored in a database.

Quality objectives	Quality characteristics	Metrics
QO.1: Transparency in amount of time	QC.1.1: Documentation of spent time for all project tasks	ME.1.1.1: Spent time per working tasks
		ME.1.1.2: Duration of individual task to the entire duration in percent
QO.2: Efficiency in maintenance	QC.2.1: Handling of customer requests	ME.2.1.1: Number of customer requests
		ME.2.1.2: Duration of processing the requests
	QC.2.2: SW – documentation	ME.2.2.1: Percentage of comments in code
		ME.2.2.1: Number and size of the in the SDP specified documents
	QC.2.3: SW – complexity	ME.2.3.1: McCabe - Factor, Function Points
QO.3: Efficiency of configuration - management	QC.3.1: Release Activities	ME.3.1.1: Number of Check-In's
	QC.3.2: Documentation of the CIs	ME.3.2.1: Percentage of documented CIs to entire number of CIs.
QO.4: Higher reuse of source code	QC.4.1: Reuse in other projects	ME.4.1.1: Number of library modules used by other projects

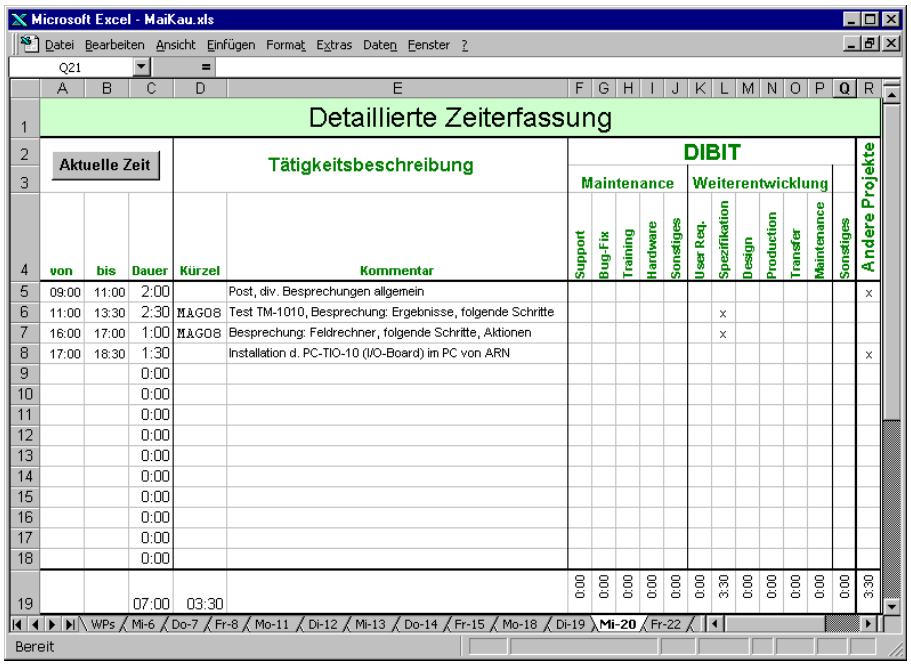
Table 4: Defined metrics

QO.2 is concerned with the efficiency of maintenance. The three quality characteristics ”Handling of customer requests”, ”SW – documentation” and ”SW – complexity” are measured by number and duration of processing customer requests, by analysing the documentation and source code comments. All customer requests just as internal detected bugs and faults are registered.

QO.3 bother the efficiency of configuration management. Quality characteristics are the intensity of use of the CCM – tool (e.g. see Fig. CGAS. 6) and the documentation of the CIs.

QO.4 defines a quality objective of higher reuse of source code. It will be measured at the end of the project by the number of library modules, which are used by other

Additionally the results of the SynQuest assessment (see Fig. CGAS. 1 and Fig. CGAS. 2) will be used as an overall metric concerned to the subjective opinion about a performed improvement.


Microsoft Excel - MaiKau.xls
Datei Bearbeiten Ansicht Einfügen Format Extras Daten Fenster ?
Q21
=
Aktuelle Zeit
WPs Mi-6 Do-7 Fr-8 Mo-11 Di-12 Mi-13 Do-14 Fr-15 Mo-18 Di-19 Mi-20 Fr-22
Bereit

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Detaillierte Zeiterfassung																	
2					Tätigkeitsbeschreibung	DIBIT												Andere Projekte
3						Maintenance					Weiterentwicklung							
4	von	bis	Dauer	Kürzel	Kommentar	Support	Bug-Fix	Training	Hardware	Sonstiges	User Req.	Spezifikation	Design	Production	Transfer	Maintenance	Sonstiges	
5	09:00	11:00	2:00		Post, div. Besprechungen allgemein													x
6	11:00	13:30	2:30	MAG08	Test TM-1010, Besprechung: Ergebnisse, folgende Schritte							x						
7	16:00	17:00	1:00	MAG08	Besprechung: Feldrechner, folgende Schritte, Aktionen							x						
8	17:00	18:30	1:30		Installation d. PC-TIO-10 (I/O-Board) im PC von ARN													x
9			0:00															
10			0:00															
11			0:00															
12			0:00															
13			0:00															
14			0:00															
15			0:00															
16			0:00															
17			0:00															
18			0:00															
19			07:00	03:30		0:00	0:00	0:00	0:00	0:00	0:00	3:30	0:00	0:00	0:00	0:00	0:00	3:30

Fig. CGAS. 5: Example of a time recording form

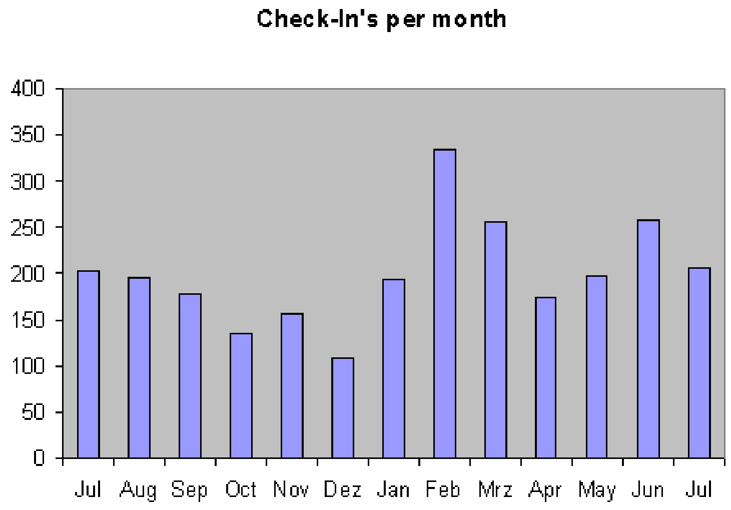


Fig. CGAS. 6: Number of Check-In's in DIBIT (ME.3.2.1)

The actual state of the measurement activities is described by intensive data gathering. The technical conditions and infrastructure were installed. The tool Provista/QS together with MKS Source Integrity supports the analysis of source-code, Releases and CIs. All data out of these tools are collected in central MS-Access databases.

Final assessment - Reporting and dissemination: The final phases of the experiment represent the Final SynQuest Assessment, which should provide a comparison to the initial assessment, and dissemination at international just as at internal events.

5. Results and Analysis

At of the experiment following results and analysis could be given.

In the moment there exists not so many quantitative data to publish, establishing the results came out from the PIE. Especially trends are – at the moment - hard to observe. Therefore the time of data gathering is to short.

Nevertheless, we recognised, that the introduction of metrics is absolutely necessary. It initiates new sights on different areas of the software developing process. And the famous sentence from Tom DeMarco [3] "You cannot control what you cannot measure." is hard to disprove.

From a technical point of view, the new tools and defined processes, have a deep influence on practical working of the employees. In some cases the benefits in daily work is obvious and gets from there a great acceptance. E.g. the CCM-Tool integration in the developing environment enhances the version management compared to earlier hand made versioning, e.g. organised in directory structures.

The CCM-Tool makes it possible for the first time to reuse software components, which is one important objectives of the PIE. Before the experiment no common software pool was installed. A lot of different versions and variants of software components were spread over local PCs and local directories on workstations. In the meantime two common libraries are in the repository (17020 LOC in 51 modules and 1495 LOC in 12 modules), which are of great interest for other projects. QO.4 will measure the reuse of such libraries in other projects.

Also the Intranet DIBIS, as information pool, has some similar benefits. Before IECS, documentation spread over different platforms and file-formats, which made it impossible to integrate them. With the HTML – standard this problem can be solved. All the advantages from an Intranet can be used under a homogeneous environment. The acceptance of DIBIS is growing and can be measured by the access count of the Internet Server.

The very common objective of shorter development cycles is also influenced in a hard measurable manner by the CCM – tool as a repository. Indirect by the now possible reuse of software – components and directly by accessing a well defined baseline stored in the repository. Before the experiment, available components had to be searched at different places and in the care of other employees.

The request database provides an easy to use and automated tool, to document all requests. It was therefore well accepted by the baseline–project staff. There existed no structured documentation or recording of customer requests or internally detected bugs in the past. The actual statistic of recorded request (e.g. 22 entries in May, 39 in June, 14 in July, 21 in August, 24 in September, 5 in October) can be used to illustrate the improvement triggered by the new introduced technologies. Before recording requests, some of them were easily forgotten and had therefore not enhanced the confidence of the customers.

The improvement of the business matters of the baseline-project was the major goal specified by the project manager of DIBIT, however a major impact on the business operation is not yet measurable.

The expenditure for installing and maintaining the CCM – tool was higher than calculated. It took some time to cope with some initial problems. An experiment with a small pilot project or better, but more expensive, the commissioning of the tool supplier to installation and training could reduce time. About 143 person days (only 60 were planned) were used for installation and training and solving initial problems.

The more detailed time recordings supports the project controlling and thereby indirect the project business results. For the quality objective QO.1 (see Table 4) the record of time spent for all project tasks is unavoidable. Before the experiment only monthly time recordings had to be prepared. For the internal use a more detailed recording scheme was installed. Based on MS-Excel a detailed time recording form had to be filled out daily (see Fig. CGAS. 5). All times spend for work packages defined in the DIBIT project had to be recorded. For some employees this was a great readjustment in their working practices, other uses this forms now for their whole time recording.

The now available common repository showed us the strength's of parts of our software. Currently we are engaged in a negotiations, where we have the opportunity to sell great parts of our image processing software as a library for a hardware manufacturer.

Support and resistance of concerned employees depends strongly on the individual. People with higher responsibilities and more experience in handling problems are more interested but also more critical than people with lower responsibilities. The greatest resistance raised at the programmer's level.

Resistance could best overcome by convincing that changes are necessary and that the initial overhead (e.g. administrative, settling into new tools) becomes smaller and the benefits of the new methods will predominate in long term. Several times there exist convincing arguments with a practical context. Therefore only examples and data in good literature can taken to argue.

It has turned out, that it is very important to include all concerned people into the decision making process. A commanding style doesn't work really today.

6. Lessons learned

The following lessons have we learned up to now. Maybe some of them will be overruled in the last time of the experiment.

Technological point of view

- Keep quality Documents as short and concise as possible! PES, CCM-Plan, etc. must be as short as possible to find acceptance by all employees. The effort of studying them must be low. Long and boring documents would not be read.
- Don't be lost in perfectionism, if some initials are not quite optimal! By introducing a new method it can sometimes happen, that also other fields

seems to need a ”face lifting”. That means, if the introduction of CCM is the focus of the experiment, you can’t also focus on an excellent introduction of a metric program, a well defined and approved software developing process and in addition an ingenious automated documentation administration system. You have only one PIE for one new process to introduce. Furthermore, that means not to completely forgot related areas.
- Consider the human being as the most important resource of the software engineering process (see [3], [16] and [2])! The best new method will not trigger the interest of dissatisfied, frustrated or slave driven employees. If there is only one professional who is not convinced of the new methods, he is able to disturb the whole process. Take more care to this employee to turn him in the same direction as the remaining team.
- Easy available information about reusable software is essential for acceptance! Without an appropriate mechanism to spread documentation of available libraries to all programmers the reusable software will not be utilised.

Business point of view

- Reusable components are investments for future! Even if the effort for design and implementation of reusable software should not be underestimated, business advantages are obvious in the long term.
- Not underestimate the expenditure for introducing a CCM-Tool! We have underestimated the expenditure for introducing a completely new technology. Next time it may be a consideration to ”outsource” the installation, administration and training on a new tool with this importance. Don’t let yourself be fooled by brochures, which promise ”easy to use”.
- Define the requirements for an external consultancy precisely! You and also the external consultant should work out the requirements and extent of the consultancy. The excellent references of a consultant are not enough to go directly into the consultation. Specify your needs and your questions and work out requirements, which must be fulfilled for acceptance of the consultation.

Strengths and weaknesses of the experiment

- We locate a weakness of the PIEs in common in the obligate dissemination events, which are hard to attend with pure practical but not scientific or innovative reports. We think a PIE is not the platform for new scientific discoveries. Instead stable and well adopted practices are introduced, which are not so interesting for the academic world. The gap between practice and theory seems to be hard to reconcile. For dissemination it seems to be profitable to exchange experiences between PIEs like in the ESPINODE program or at EPIC.
- Introduction of new techniques was usually done on the fly. Using a separate project (e.g. a PIE) is a much more powerful method to introduce something new.

Common lessons learnt

- Take more frequently a look into the Internet for good information as well as for managing the ESSI-PIE (e.g. VASIE - or ESSI homepage, with good hints and disseminated information from other PIEs)! In evaluating software tools excellent comparison reports could be found.
- Concentrate on the baseline project and don't try to introduce new technologies to the whole organisation! Especially in the earlier phases of the PIE, don't forget that the experiment should be carried out on one baseline project. You can focus to the objectives and only to a limited number of different opinions. This is important, if you have to convince people for new technologies.

7. Conclusions and Future Actions

The introduction of CCM as a supporting process for the software developing process in the course of the ESSI PIE IECS is in a final state. The baseline project is in a busy phase of maintenance and distributes a lot of data for measuring. The new processes are in use and are well accepted by the project staff.

Without listing quantitative data, the most objectives seems to be achieved. Especially the software reuse and the configuration and change management goals are obviously achieved. Also documentation and the higher degree of reproducibility and readability of code has been improved. Most of the concerned employees are high motivated and very interested in the new methods and technologies.

Many of the processes introduced in the first half of the PIE have to be consolidated during the second half of the experiment. Especially the documentation of the new procedures must be improved. Precise, but if possible not voluminous descriptions of procedures, templates and checklists shall make it possible to support others than the baseline project, which stands in a privileged position to the PIE.

The higher degree of automation of the metric data gathering process, but also the exploitation and interpretation of the gathered data must be improved.

Glossary

CCM	**C**hange and **C**onfiguration **M**anagement
CI	**C**onfiguration **I**tem
DIB	German abbreviation for the department Digital Image Processing: Institute für **Di**gitale **B**ildverarbeitung
DIBIS	Abbreviation for the DIB Intranet: **DIB I**nformation **S**ystem
DIBIT	Abbreviation for the baseline project: **Di**gital Image O**b**servat**i**on System for **T**unnelling
IECS	**I**mprovement of **E**fficiency by introduction of **C**CM and **S**oftware engineering standards
ME	**M**etrics

PES	**P**roject **E**ngineering **S**tandards
QC	**Q**uality **C**haracteristic
QO	**Q**uality **O**bjective
SDP	**S**oftware **D**eveloping **P**rocess
SME	**S**mall and **M**edium sized **E**nterprise
SQG	**S**oftware **Q**uality **G**roup

Appendix I: About the authors

Clemens Gasser is a staff member of 3D Vision Group and some other project groups at the JOANNEUM RESEARCH Industrial Analysis department in the Institute of Digital Image Processing (DIB). He is Diploma Engineer in Telematics (University of Technology Graz) on studies in the field of artificial intelligence (neural networks, fuzzy reasoning and control) and industrial image analysis. He has collected experience on DSP, parallel computing, software engineering and software quality. In 1995 Clemens Gasser received his Diploma with thesis on intelligence adaptive fuzzy control systems. Since May 1995 he is a fixed employee at JOANNEUM RESEARCH. Between 1995 – 1997 he leads and assists in different industrial image processing projects. He was a staff member of the project team, which developed a system for stereo-vision based tunnel surface reconstruction (DIBIT). Since March 1997 he is the project manager of the ESSI-PIE IECS. This ESPRIT - project (Nr. 23760) experiments with the Introduction of CCM into the software developing cycle. Since September 1998 he works as internal auditor of the ISO 9001 QM system.

Edwin Deutschl is a staff member of the Linescan Vision Group at the JOANNEUM RESEARCH Industrial Analysis department in the Institute of Digital Image Processing (DIB). He studied Applied Mathematics at the University of Technology in Graz, Austria. In 1994 he received his Diploma with a thesis on light reflections. Since 1995 he works at the JOANNEUM RESEARCH Center in Graz as application developer and project manager of Online Quality Assurance Systems. His research interests are in real time and parallel computing for applications in wood- and pharmaceutical industry as well as related Software Engineering issues. Since 1997 he is a permanent member of the Software Quality Group at the Institute of Digital Image Processing.

Appendix II: About JOANNEUM RESEARCH

JOANNEUM RESEARCH is one of the largest technology centres in central Europe, making its expertise available to business, industry and administration. Our highly-qualified staff of 300 people that work in 20 research units implement their know-how in all sectors of innovation, both at national and in international levels. Our service includes specifically geared development tasks for small- and medium-sized companies, complex interdisciplinary national and international research assignments as well as tailored techno-economic consulting. Our highest priority in all our activities is to meet the top quality standards demanded by our clients.

References

[1] Babich, Wayne A.: Software Configuration Management, Addison – Wesley, 1986

[2] Brooks F.: The Mythical Man-Month. Reading, MA, Addison Wesley, 1975

[3] DeMarco T., Lister T.: Wien wartet auf Dich! Der Faktor Mensch im DV-Management. München, Wien; Hanser, 1991

[4] DeMarco T.: Structured Analysis and System Specification, Englewood Cliffs, NJ, Yourdon Press/Prentice Hall, 1978.

[5] DISC TickIT Office: The TickIT Guide - A Guide to Software Quality Management System Construction and Certification to ISO 9001, London, ISBN 0 580 25107 1, 1995.

[6] El Emam K., Drouin J. N., Melo W.: SPICE - The Theory and Practice of SPI and Capability Determination, IEEE Computer Society, 1998.

[7] Paar G., Kuijpers N., Gasser C.: Stereo Vision and 3D Reconstruction on a Processor Network. Machine Perception Applications, Graz, September 2-3 1996. IAPR/TC8.

[8] Humphrey, Watts S.: Managing the Software Process. Reading, MA, Addison Wesley, 1989.

[9] ISO 9000-3, 1.Ausgabe, 1991-06-01, Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software. 1991.

[10] Kehoe R., Jarvis A.: ISO 9000-3 - A tool for software product and Process Improvement, Springer-Verlag New York, 1996.

[11] Mortice Kern System Inc., Canada: http://www.mks.com.

[12] Möller K.H., Paulish D. J.: Software Metrics - A practitioner's guide to improved product development.

[13] Muelleitner G., Steinmann C.: Minimum Metrics for Software Production. IIG-Report 395, TU Graz 1994.

[14] Sanders J., Curran E.: Software Quality - A Framework for Success in SW- Development and Support. Addison Wesley, ISBN 0-201-63198-9, 1995.

[15] http://www.cs.colorado.edu/homes/andre/public_html/configuration_management.html

[16] Yourdon E.: Die Westliche Programmierkunst am Scheideweg - die Schlüsseltechniken der SW-Eentwicklung für das 21. Jahrhundert. Hauser Verlag, 1993