Proceedings the
EuroSPI 2000 Conference

EuroSPI

European Software Process Improvement

Theme 2000:
"Practical and Innovation Based Software Process Improvement to Prepare for the New Millenium"

7-9 November 2000
Copenhagen Business School, Denmark

CBS
Copenhagen Business School

COPENHAGEN 2000

ISBN 952-9607-29-6

Digital tryk: Klink Offset, Herlev - EMAS reg. DK-S-0083

Foreword EuroSPI 2000

This is the 7th European Software Process Improvement conference (EuroSPI) of a series of conferences to which countries from across Europe the rest of the world contribute their lessons learned and share their knowledge of how to reach the next level of software maturity. The EuroSPI conferences present and discuss practical results from improvement projects in the industry, focussing on the obtained benefits and the criteria for success. Leading edge European industry contribute to and participate in this event. This year's event is the third to be hosted in Scandinavia which has one of the most vibrant SPI communities in Europe and is home to some of the top electronics companies.
The theme for this years conference is "Practical and Innovation Based Software Process Improvement to Prepare for the New Millennium". We have selected and grouped the papers into 12 sessions with headings indicating the fundamental content of the papers:

- SPI and Assessment
- SPI and People / Skills
- SPI and Implementation
- SPI and Systems
- SPI and Personal Processes
- SPI and Measurement I
- SPI and Measurement II
- SPI and Testing
- SPI in Small Businesses
- SPI and Procurement
- SPI and Requirements
- Danish Experiences in SPI

Two distinguished scientists will provide key-note presentations: Watts S. Humphrey from Software Engineering Institute (SEI), USA and Dieter Rombach from Frauenhofer IESE, Germany. Watts Humphrey will talk about "What if your Life Depended on Software?" and Dieter Rombach will make his presentation under the title "Experiences with Measurement and Goal Based Strategies". We believe we all have something wise, thought-provoking, and enjoyable to look forward to in these two key-note presentations.

A special additional track is based on the Danish initiative "Center for Software Process Improvement". This project included 4 software organisations, Brüel & Kjær A/S, Danske Data A/S, L. M. Ericsson Denmark A/S, and Systematic Software Engineering A/S, and more than a dozen researchers from Aalborg University, Technical University of Denmark (DTU), and DELTA Danish Electronics, Light & Acoustics. This national initiative was funded by the Danish government (Ministry of Commerce (Council for Development of Business and Industry) and Ministry of Research (Centre for IT-research)) and the participants. The total budget was 2.6 mill Euro with a duration of 3 years (from 1997 to 2000) involving 35 people directly in more than 15 SPI activities. The final results are to be published in a book with the title "Learning to Improve"; an anthology with 18 articles describing the lessons learned during the project. At EuroSPI 2000, we have the opportunity to present 7 of the articles from the book.

A special thank is to the Copenhagen Business School for hosting the EuroSPI 2000 conference. The new building is designed by the Danish company Wilhelm Lauritzen AS Architects and officially opened in 1999. We are looking forward to the 3 days in these exciting surroundings.

Last but not least we wish to thank the authors for their effort in preparing their contributions. We have enjoyed planning the programme for EuroSPI 2000 and we hope you will all find you visit to Copenhagen and Denmark enjoyable and worthwhile.

Bernd Hindel, 3Soft
Carsten Jørgensen, DELTA
John Elliot, DERA
Mads Christiansen, DELTA
Richard Messnarz, ISCN
Risto Nevalainen, STTF
Tor Stålhane, SINTEF
Yingxu Wang, IVF

Contents:

Conference Board

International Organiser

Local Organiser

Session 1- SPI and Assessments / Benchmarks

Session Chair:

Tor Stalhane,

Sintef

A Recent Extension of ISO 15504 to IT Acquisition Processes

Yingxu Wang
Dept. of Electrical and Computer Engineering
University of Calgary
2500 University Drive, Calgary, Alberta, Canada T2N 1N4

IVF Centre for Software Engineering
Argongatan 30, S-431 53, Molndal, Gothenburg, Sweden
Yingxu.Wang@acm.org

Abstract: A recent extension of ISO/IEC 15504 (SPICE) to cover IT acquisition processes is proposed and implemented by a European research project PULSE. This paper describes the PULSE process model and process assessment method within the context of IT acquisition process assessment and improvement. The PULSE methodology is featured by a tool-based process assessment method, a template-based assessment data collection method, and a prototype for assessment report generation. Industry trials have shown the usefulness and impact of the PULSE model and related assessment results on improving IT system and product acquisition processes in a wide range of industry sectors.

Key words: Software engineering, ISO 15504, extended model, IT acquisition process, assessment, methodology, tool, industry trial

1. Introduction

The PULSE project intends to improve the IT acquisition processes of organisations involved in the acquisition of IT products and systems containing software. The project combines two approaches to assist such organisations:

- firstly by defining and verifying a formal methodology for identifying and assessing the processes used by such organisations for IT acquisition;
- secondly by identifying a set of organisational actions that improve the ways in which acquisitions are managed and the measurement of the success of IT acquisition team.

By combining both formal methods and organisational techniques, the PULSE project expects to provide an extremely broad set of actions to assist organisations in improving the way that they acquire IT products and systems.

The PULSE project develops a set of methodology and models covering:

- an IT acquisition reference model for processes and process capability [1];
- an IT acquisition assessment model [2];
- an IT acquisition assessment method [3];
- an IT acquisition assessment tool [4];
- a training syllabus for IT acquisition assessors [8];
- a certification scheme for IT acquisition assessors [9].

The PULSE IT acquisition process assessment methodology is designed to comply with the general requirements for an extended compatible model as defined in ISO/IEC TR 15504-2 [10] and for performing 15504-compliant assessments as defined in ISO/IEC TR 15504-3 [11]. The PULSE IT acquisition processes have been formally accepted by ISO/IEC JTC1/SC7 as an officially extended model for ISO/IEC 15504 (SPICE) [12, 13].

2. Structure of PULSE IT Acquisition Process Models

The PULSE methodology defines the PULSE IT acquisition reference model [1] for process and process capability, assessment model [2], assessment method [3] and assessment tool [4]. In addition, a training syllabus [8] and a certification scheme [9] are available so that potential assessors may be taught how to apply the methodology consistently.

The context and inter-relationship of the various components in the PULSE methodology is illustrated in Fig.1.

The reference model is a generic model – this means it may be applied in a large population of acquisition types from standalone to total systems, from small to progressive acquisitions, from new to enhancements or modifications of existing systems.

For acquisition process assessment, an assessor uses a more detailed assessment model compatible with this reference model, containing a comprehensive set of indicators of process performance and process capability, to make judgement about the capability of the processes. This may be achieved in a particular context of acquisition, permitting judgement about the comprehensiveness of the acquisition processes and their acquisition capabilities.

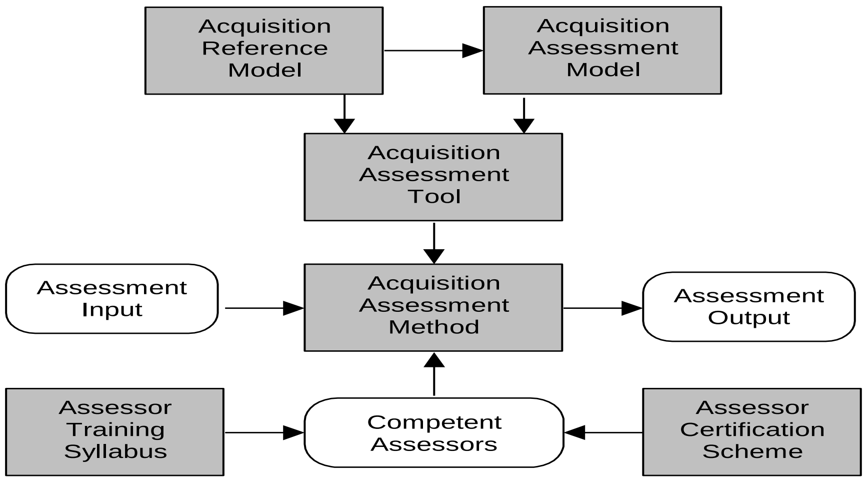

Acquisition Reference Model
Acquisition Assessment Model
Acquisition Assessment Tool
Assessment Input
Acquisition Assessment Method
Assessment Output
Assessor Training Syllabus
Competent Assessors
Assessor Certification Scheme

Figure 1. Structure of the PULSE IT acquisition process models

The assessment method describes how an assessment shall be performed in a step by step manner with the support of an assessment tool and a set of templates. The assessment method is designed to incorporate the phases, processes and component-processes of an assessment, and to regulate the format and contents of the output work products produced.

3. The PULSE Process Reference Model

A reference model of PULSE is developed [1] for IT acquisition processes and process capability that forms the basis for IT acquisition process assessment. The reference model defines, at a high level, the fundamental objectives that are essential to good IT acquisition. The high-level objectives describe what is to be achieved, not how to achieve them.

This reference model is applicable to any organisation wishing to establish and subsequently improve its capabilities in the acquisition of IT products. The model does not presume particular organisational structures, management philosophies, life cycle models, technologies, or methodologies.

The IT acquisition reference model is a generic model – this means it may be applied in a large population of acquisition types from standalone to total systems, from small to progressive acquisitions, from new to enhancements or modifications of existing systems. Acquisition can be triggered at any point – for new products, enhancements, or during operational support of a system. This model caters for all triggers from total systems acquisition to components of total systems or standalone components. The acquisition products may evolve progressively – the model can accommodate evolution of acquisition requirements. In its specific context, the details of its implementation and work products may vary.

The architecture of this reference model organises the processes to help acquisition personnel understand and use them for continuous improvement of the management of acquisition processes. Used with team and organisational focus, it offers a sound approach manage, improve and increase the capability of an organisation in acquiring the right systems the right way.

The reference model architecture is made up of two dimensions:

- The *process dimension*, which is characterised by process purpose statements which are the essential measurable objectives of a process;

- The *process capability dimension*, which is characterised by a series of process attributes, applicable to any process, which represent measurable characteristics necessary to manage a process and improve its capability to perform.

3.1 The Process Dimension

In the process dimension, the reference model categorises 25 processes and 16 component-processes into three life-cycle process groupings which contain four process categories, according to the type of activity they address.

A structure of the process system is described in Table 1, where a component-process is a sub-process which defines independent activities and functions in the theme of a process.

The PULSE process categories and processes provide a grouping by type of activity. Each process in the reference model is described in terms of a purpose statement. These statements comprise the unique functional objectives of the process when instantiated in a particular environment. The purpose statement includes additional material identifying the outcomes of successful implementation of the process. Satisfying the purpose of a process represents the first step in building process capability.

An example of a defined process is shown below:

5.1.1.5 ACQ.2 Requirements Definition

The purpose of the *Requirements Definition process* is to establish the requirements of the acquisition. Successful implementation will gather, define, and track current and evolving acquisition needs and requirements throughout the life of the contract so as to establish successive acquisition requirement baselines. As a result of successful implementation of the process:

- the different perspectives (e.g. financial, contractual, technical, project) of acquisition requirements that meet the needs of the acquirer will be defined,
- the requirements will be revised to remain consistent with acquisition,
- the requirements and potential solutions will be communicated to the affected groups,
- new or changed requirements will be incorporated into the requirements baseline.

Table 1. PULSE process structure

Symbol	Category	Process	Component-process
ACQ	Acquisition		
ACQ.1		Acquisition needs	
ACQ.1.1			Acquisition policy
ACQ.1.2			Acquisition strategy
ACQ.1.3			Benefits analysis
ACQ.2		Requirements definition	
ACQ.2.1			Technical requirements
ACQ.2.2			Contract requirements
ACQ.2.3			Financial requirements
ACQ.2.4			Project requirements
ACQ.3		Contract award	
ACQ.3.1			Invitation to tender
ACQ.3.2			Tender evaluation

ACQ.3.3			Contract negotiation
ACQ.4		Contract performance	
ACQ.4.1			Supplier monitoring
ACQ.4.2			Acquisition acceptance
ACQ.4.3			Contract closure
SUP	Support		
SUP.1		Documentation	
SUP.2		Configuration management	
SUP.3		Quality assurance	
SUP.4		Verification	
SUP.5		Validation	
SUP.6		Joint review	
SUP.7		Audit	
SUP.8		Problem resolution	
MAN	Management		
MAN.1		Management	
MAN.2		Project management	
MAN.3		Quality management	
MAN.4		Risk management	
ORG	Organisation		
ORG.1		Organisational alignment	
ORG.2		Improvement	
ORG.2.1			Process establishment
ORG.2.2			Process assessment
ORG.2.3			Process improvement
ORG.3		Human resource Management	
ORG.4		Infrastructure	
ORG.5		Measurement	
ORG.6		Reuse	
ORG.7		Financial management	
ORG.8		Manage supplier relationships	
ORG.9		Manage user relationships	

3.2 The Process Capability Dimension

Evolving process capability is expressed in terms of process attributes grouped into capability levels. Process attributes are features of a process that can be evaluated on a scale of achievement, providing a measure of the capability of the process. Process attributes are applicable to all processes. Each process attribute describes a facet of the overall capability of managing and improving the effectiveness of a process in achieving its purpose and contributing to the business goals of the organisation.

A capability level is characterised by a set of attributes that work together to provide a major enhancement in the capability to perform a process. Each capability level provides a major enhancement of capability in the performance of a process. The levels constitute a rational way of progressing through improvement of the capability of any process.

The process capability dimension of the PULSE reference model [1] is aligned to and is identical to the capability dimension of ISO/IEC TR 15504 Part 2 [10]. The reference model alone is not intended to be used as the basis for conducting reliable and consistent assessments of process capability since the level of detail is not sufficient. The reference model needs to be supported with a comprehensive set of indicators of process performance and capability. In this way, consistent ratings of process capability will be possible. An assessment model [2] that incorporates such a set of indicators is provided in Section 4.

4. The PULSE Process Assessment Model

For a PULSE acquisition process assessment, an assessor uses an assessment model [2], compatible through mapping and data transform mechanisms to the PULSE reference model.

The assessment model contains a comprehensive set of indicators of process performance and process capability, to make judgements about the capability of the organisation's processes. This may be achieved in a particular context of acquisition, permitting judgement about the comprehensiveness of their acquisition processes and their acquisition capabilities.

This section defines the PULSE assessment model that is compatible with the reference model [1]. The assessment model defines the overall structure, terminology and compatibility of the assessment model with the reference model. Attribute indicators, which are used to support assessment, are embodied within the assessment tool [4] and are referenced within the assessment model as assessment tool supporting documents [5, 6]. The indicators are used as guides in collecting the objective evidence that enables an assessor to assign ratings to process attributes.

4.1 Structure of PULSE Assessment Model

Figure 2 shows the structure of the assessment model and relationship with other components in the PULSE methodology. The assessment model architecture consists of the process and capability dimensions. The process dimension of the assessment model is identical to the process dimension of the reference model. The capability dimension of the assessment model however is different to the capability dimension of the reference model. The relationship between the two is achieved by a mapping between the key elements of the two models and through a capability transformation algorithm, which is described in Section 6.

In the process dimension, each process has a defined set of associated input and output work products (WP) which may be evidenced when judging performance of the process. The work products are identified with reference to generic work products (GWP) definitions which themselves have generic work product characteristics. The associated work products for each process are contained in [5], which provides the foundation for incorporating process input and output work product definitions within the assessment tool.

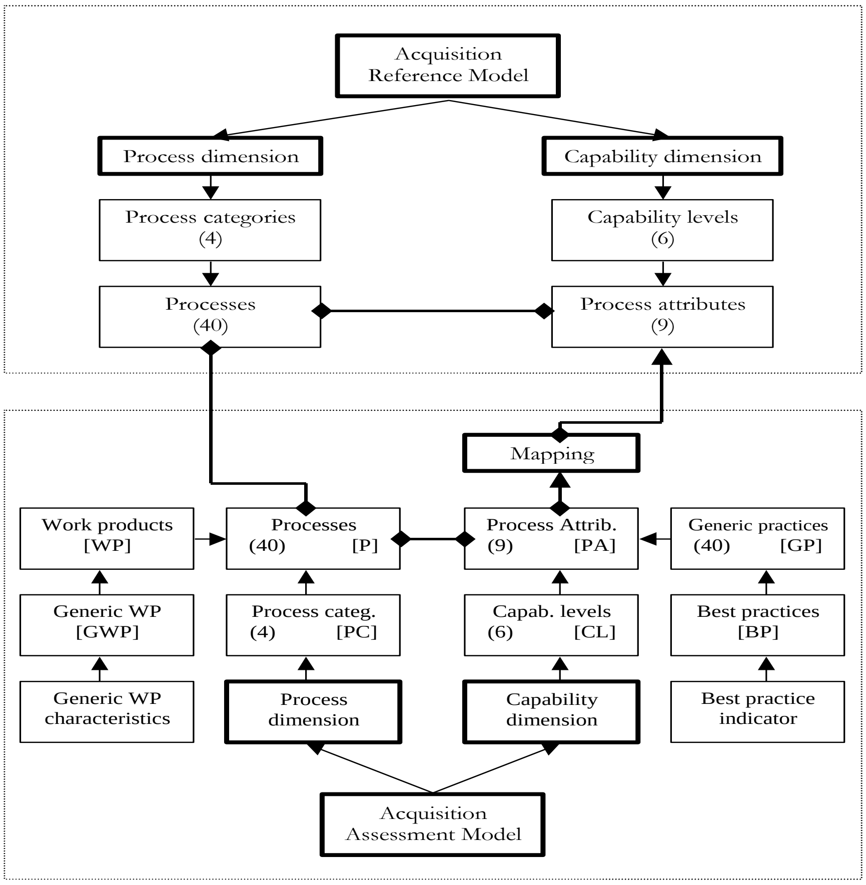


Figure 2. Structure of the PULSE assessment model

Each process attribute (PA) has associated with it a number of generic practices (GP) [2]. The generic practices are universal activities and are intended to be applicable to all processes. They are designed around the achievement of the principal management functions of planning, organising, resourcing and controlling.

Generic practices are means for achieving the capabilities addressed by process attributes. Evidence of generic practice performance supports the judgement of the degree of achievement of the process attribute. A set of unique generic practices are linked to each process attribute. The sets of generic practices are intended to be applicable to all processes in the process dimension of the model.

Associated with each process attribute for each and every process are defined a number of best practices (BP) and best practice indicators [6] extracted from world-wide industry best practice. These practices have then been grouped according to the relevant process attributes and then indicators have been determined for these best practices.

The generic practices, best practices and best practice indicators together with the input/output work products and their characteristics all represent types of evidence that would substantiate judgements of the extent to which a process attribute is being achieved.

4.2 Process Dimension of the PULSE Assessment Model

The process dimension of the assessment model is identical to that of the reference model. Process categories and processes provide a means of grouping by type of activity. Each process in the assessment model, as well as in the reference model, is described in terms of a purpose statement. These statements comprise the unique functional objectives of the process when instantiated in a particular environment. The purpose statement includes additional material identifying the outcomes of successful implementation of the process. Satisfying the purpose of a process represents the first step in building process capability.

4.3 Capability Dimension of the PULSE Assessment Model

Evolving process capability is expressed in the assessment model in term of process attributes grouped into capability levels. Process attributes are features of a process that can be evaluated on a scale of achievement, providing a measure of the capability of the process. They are applicable to all processes. Each process attribute describes a facet of the overall capability of managing and improving the effectiveness of a process in achieving its purpose.

A capability level is a set of process attribute(s) that work together to provide a major enhancement in the capability to perform a process. Each level provides a major enhancement of capability in the performance of a process. The process capability levels constitute a rational way of progressing through improvement of the capability of any processes.

4.3.1 PULSE capability levels and process attributes

There are six capability levels in the PULSE assessment model, incorporating nine process attributes. Process attributes are used to determine whether a process has reached a given capability. Each attribute measures a particular aspect of the process capability. At each level there is no ordering between the process attributes.

The process attributes are evaluated on a four point ordinal scale of achievement, according to the rating scale defined in Section 6. The process attributes provide insight into the specific aspects of process capability required to support process capability determination and improvement. A list of process attributes is shown in Table 2, which are grouped to the relevant capability levels within the assessment model.

A process attribute in PULSE represents a measurable characteristic of any process. The rating scale is a percentage scale from zero to one hundred percent that represents the extent of achievement of the attribute. Usually, the PULSE process attribute, according to the PULSE assessment model [2], is rated by a scale *of fully*, *largely*, *partially* and *not* achievement.

Table 2. PULSE capability levels and process attributes

Capability Levels	Process Attributes	Description
CL0		Incomplete process
CL1		Performed process
	PA 1.1	Performed attribute
CL2		Managed process
	PA 2.1	Planed and tracked attribute
	PA 2.2	Document, configuration and change control attribute
CL3		Defined process
	PA 3.1	Process definition attribute
	PA 3.2	Quality achievement attribute
CL4		Established process
	PA 4.1	Stability attribute
	PA 4.2	Skills, competencies and training attribute
CL5		Optimised process
	PA 5.1	Technical infrastructure attribute
	PA 5.2	Efficiency and effectiveness attribute

4.3.2 PULSE capability level rating

The PULSE process capability level rating method, according to the PULSE assessment model [2], is defined in Table 3. The basic algorithm is that a capability level, *n,* will not be achieved until level *n-1* is largely or fully achieved, and the other lower levels (if any) are fully achieved.

Table 3. PULSE capability model

Scale	Description	Process Attributes	Rating
Level 1	Performed	Performed	Largely or fully
Level 2	Managed	Performed	Fully
		Planned and tracked	Largely or fully
		Document, configuration and change control	Largely or fully
Level 3	Defined	Performed	Fully
		Planned and tracked	Fully
		Document, configuration and change control	Fully
		Process definition	Largely or fully
		Quality achievement	Largely or fully
Level 4	Established	Performed	Fully
		Planned and tracked	Fully
		Document, configuration and change control	Fully
		Process definition	Fully
		Quality achievement	Fully
		Stability	Largely or fully
		Skills, competencies and training	Largely or fully
Level 5	Optimised	Performed	Fully
		Planned and tracked	Fully
		Document, configuration and change control	Fully
		Process definition	Fully
		Quality achievement	Fully
		Stability	Fully
		Skills, competencies and training	Fully
		Technical infrastructure	Largely or fully
		Efficiency and effectiveness	Largely or fully

5. The PULSE Process Assessment Method

The PULSE IT acquisition process assessment method [3] describes the usage of the assessment model and reference model in the overall assessment procedure. The assessment method is described in a number of phases, each which contains assessment method processes and component-processes. This section describes the structure of the assessment method and general approach of a PULSE assessment.

5.1 Structure of PULSE Assessment Method

The PULSE assessment method consists of 3 phases, 7 processes and 10 component-processes. A structural view of the PULSE assessment method is shown in Table 4.

Each process and/or component-process shown in Table 4 is defined in terms of its purpose, input/output work products, the method employed and templates used for data collection and logging. Additional notes are provided where relevant.

Table 4. Structure of PULSE assessment method

Phase	Process	Component-process
Assessment Input		
	Assessment input definition	
		Define assessment purpose
		Define acquisition requirement
		Define assessment scope
Assessment		
	Assessment preparation	
		Appoint assessment team
		Prepare assessment confidentiality agreement
		Plan schedule and resources
		Determine assessment reference model, assessment model and tool
		Map organisational unit processes
		Define processes to be assessed and target capability levels
		Develop assessment brief
	Data collection, validation and rating	
	Derive ratings and capability profile	
	Strengths and weakness analysis	
	Improvement opportunities analysis	
Assessment output		
	Assessment report	

5.2 General Approach of PULSE Assessment

The PULSE assessment is carried out using the support of a software tool [4] and a set of templates [3]. The assessment approach is via round-table meeting and ratings achieved by negotiation of process capabilities between assessors and assessees.

The records of a performed assessment consist of filled-in templates and data file produced from assessment software tool, which contains: ratings, evidence, notes, and various generated process capability profiles.

When multiple projects are assessed in an organisational unit, separate records where relevant will be produced for each project.

A brief description of the PULSE assessment method is presented in Table 5, where the key assessment activities are grouped by phases, processes, component-processes, with relevant records produced indicated.

Table 5. Summary of PULSE assessment method

Phase / Process	Component-process	Output work product
Assessment input		
Assessment input definition		
	Define assessment purpose	T.1 Assessment purpose
	Define acquisition Requirement	T.2 Acquisition requirement
	Define assessment scope	T.3 Assessment scope
Assessment		
Assessment preparation		
	Appoint assessment team	T.4 Assessment Team and Responsibilities
	Prepare assessment confidentiality agreement	T.5 Assessment Confidentiality Agreement
	Plan schedule and resources	T.6 Assessment Schedule and Resources
	Determine assessment reference model, processes model and tool	
	Map organisational unit processes	T.7(1) Processes To be Assessed And target Capability Levels
	Define processes to be assessed and target capability levels	T.7(2) Processes To be Assessed And target Capability Levels
	Develop assessment brief	T.8 Assessment Brief
Data collection, validation and rating		The electronic data record file of PULSE tool
Derive ratings and capability profile		Process capability rating records as produced by PULSE tool
Strengths and weaknesses analysis		T.9 Process Strengths And Weaknesses Analysis
Improvement opportunities analysis		T.10 Process Improvement Opportunities Analysis
Assessment output		
Assessment report		T.11 Assessment Report

6. Conclusions

This paper has reported a systematical work on developing a compatible assessment model for ISO/IEC TR 15504, in order to extend the emerging international standard to cover the IT acquisition processes. A complete methodology including a reference model, assessment model, assessment method, and assessment tool has been developed in the PULSE project.

The PULSE model and methodology are featured by a tool-based process assessment method, a template-based assessment data collection method, and a prototype for assessment report generation. Industry trials have shown usefulness and impact of the PULSE methodology on improving the IT system and product acquisition processes in a wide range of industry sectors. The PULSE IT acquisition processes have been formally accepted by ISO/IEC JTC1/SC7 as an official extension model for ISO/IEC 15504 (SPICE).

Acknowledgements

This project was partially funded by the European Commission through the SPRITE S2 Programme Project No. 97/501155. The author would like to thank the support of European

Commission, all partner organisations of the project, and the user organisations that participated in the industry trial of the model.

References

[1] Dorling, A. and Song M. (1998), PULSE D1.1 – *IT Acquisition Reference Model for Processes and Process Capability*, Technical Report of the European PULSE Project, pp.1-37.

[2] Dorling, A. and Wang, Y. (1998), PULSE D2.1 – *IT Acquisition Process Assessment Mode*l, Technical Report of the European PULSE Project, pp.1-29.

[3] Wang, Y. and Dorling, A. (1998), PULSE D3.1 – *IT Acquisition Process Assessment Method,* Technical Report of the European PULSE Project, pp.1-61.

[4] Dorling, A., Wang, Y. and Steinmann, C. (1998), PULSE D.4.2 – *IT Acquisition Process Assessment Tool*, Technical Report of the European PULSE Project,

[5] Pitette, G.m. (1998), PULSE D.4.2.1 – *IT Acquisition Processes and Associated Work Products*, Technical Report of the European PULSE Project, pp.1-25.

[6] Kirchhoff, U. and Sundmaeker, H. (1998), PULSE D.4.2.2 – *IT Acquisition Process Attribute Best Practices and Best Practices Indicator Set*, Technical Report of the European PULSE Project, pp.1-99.

[7] Wang, Y. and Dorling, A. (1998), PULSE D4.4 – *IT Acquisition Assessment Report Template*, Technical Report of the European PULSE Project, pp.1-12.

[8] Wang, Y. and Dorling, A. (1998), PULSE D6.1 – *IT Acquisition Process Assessor Training Curriculum*, Technical Report of the European PULSE Project, pp.1-15.

[9] Dorling, A. (1998), PULSE D6.2 – *IT Acquisition Assessor Certification Scheme*, Technical Report of the European PULSE Project, Technical Report of the European PULSE Project, pp.1-10.

[10] ISO/IEC 15504-2 (1999), Information Technology - Software Process Assessment Part 2: *A Reference Model for Processes and Process Capability*, pp.1-39.

[11] ISO/IEC 15504-3 (1999), Information Technology - Software Process Assessment Part 3: *Performing an Assessment*, pp.1-7.

[12] Dorling, A., Wang, Y., et al. (1999), Reference Model Extensions to ISO/IEC TR 15504-2 for Acquirer Processes, ISO/IEC JTC1/SC7/WG10, Curitiba, Brazil, May, pp. 1-34.

[13] Dorling, A., Wang, Y., Kirchhoff, U., Sundmaeker, H., Maupetit, C., Pitette, G., Pereira, J. and Hansen, S. (1999), *ICT Acquisition Process Assessment Methodology,* The PULSE Consortium, March, pp.1-87.

About the Authors

Yingxu Wang is Professor of Software Engineering in Dept. of Electrical and Computer Engineering at The University of Calgary, and project manager with the Center for Software Engineering at IVF, Gothenburg, Sweden. He received a PhD in software engineering from the Nottingham Trent University / Southampton Institute, UK. He is a member of IEEE, ACM, and ISO/IEC JTC1/SC7, and is Chairman of the Computer Chapter of the IEEE Swedish Section. He was a visiting professor at Oxford University. He is the lead author of a recent book on *Software Engineering Processes: Principles and Applications*, and he has published over 100 papers.

A Detailed Process Assessment Method For Software SMEs

Timo Mäkinen & Timo Varkoi
Tampere University of Technology,
Information Technology, Pori, Finland
Marion Lepasaar
Tallinn Technical University, Estonia

1. Abstract

Small software organisations starting to develop their software processes require detailed information about what to improve. This information can be obtained by using process assessments. In the beginning the emphasis is on software process improvement (SPI) instead of capability evaluation. The high abstraction level of light assessment methods does not provide adequately detailed information for process improvement planning and actions. On the other hand, proper use of heavy methods can be too complicated for small organisations with limited resources.

This paper presents a method for small organisations to conduct the assessments and to collect process improvement ideas in an effective manner. The goal is to provide a method that is easy to use but yields detailed improvement actions. The assessment process and its work products are tailored according to the needs of small companies. The method is compatible with SPICE and has been tested in several assessments.

This method utilises detailed indicators to generate a variety of improvement ideas. Some of the ideas and the candidate improvement actions arise in discussions with experts during thorough process assessment sessions. Additional improvement actions are derived from the indicator ratings of the assessed processes. A detailed assessment report with multi-level conclusions refines the assessment findings and supports the improvement planning and activities over a period of time.

Keywords: software process improvement, process assessment methodology

2. Introduction

Ideally the SPI activities should be aligned with the organisation's business goals and stated in their business strategy. At the beginning of the SPI effort the maturity of small organisations is usually quite low. For instance, written strategies can be missing.

SataSPIN is a project started in August 1998 to establish a software process improvement network (SPIN) in the Satakunta region in Western Finland. The core of the project is to help small and medium sized enterprises (SMEs) in the software business to develop their operations using international software process models. In the first phase of the project eight small software organisations are participating to improve their software processes. [7]

The SataSPIN project tailored the SPI initiation phases according to the needs of the participating organisations [6, 8]. The SPI initiation framework in fig. TMAKINEN.1 covers the process improvement start-up from the examination of an organisation's needs to the development and support of the SPI programme plan. The framework consists of three steps. First, the organisation needs to understand the possibilities of SPI in achieving its business goals. Secondly, software processes are assessed in a reliable manner. And thirdly, the SPI activities need to be planned and supported.

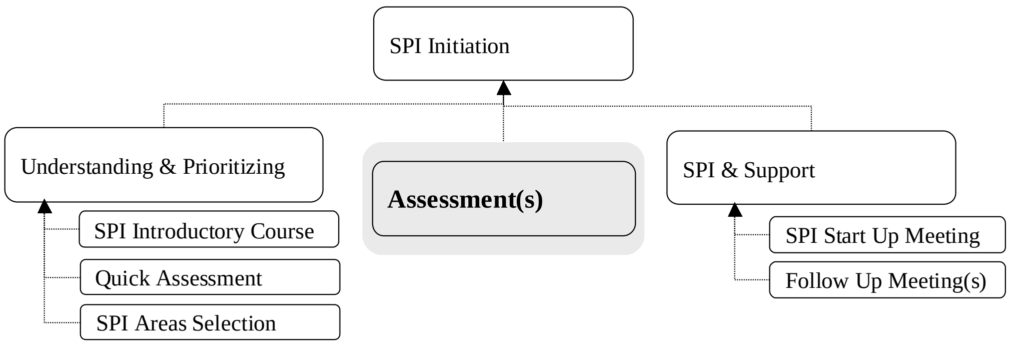


Fig. TMAKINEN.1 : SPI initiation framework

This paper concentrates on the method used in assessments. The first chapter describes the assessment model behind the method: the SPICE assessment model is extended with features developed by the Finnish Software Metrics Association (FiSMA) and the SataSPIN project itself. The second chapter presents the assessment process: activities, work products and tools. The third chapter is about the assessment report. The presentation is concluded with a discussion of the method's benefits and future work.

3. The assessment model

SPICE (Software Process Improvement and Capability dEtermination) was developed by ISO and is documented as ISO 15504 Technical Report containing nine parts. The normative reference model for software process assessment is defined in SPICE Part 2 [2]. The two-dimensional model of SPICE consists of a set of processes and a set of process attributes. The process attributes apply across all processes and are grouped into five capability levels that may be used to determine the capability of the process. The processes themselves are grouped into five process categories. SPICE also presents an exemplar assessment model in Part 5 [3], which extends the process model with assessment indicators such as base and management practices. [4]

The level 1 assessment indicators of our SPICE compatible assessment model are shown in fig. TMAKINEN.2. Processes and process outcomes are defined in SPICE Part 2. It is possible to assess a process by rating only the achievement of the process outcomes. Usually our assessments are based on base practice ratings. Base practices (BP) of the processes are presented in SPICE Part 5. The model has been tailored by FiSMA to contain detailed base practice indicators that serve as a checklist in an assessment. In addition the model supports the checking of work products. Both of these two additional indicators make it easier to provide evidence for the process capability rating.

The base practice indicators provide evidence and make rating easier. For instance, the indicators for a base practice of the Requirements elicitation process, CUS.3.BP2 Agree on requirements, are:

- Customer requirements are available e.g. attached to offers and contracts
- The project group is aware of and understands the requirements
- All stakeholders, e.g. parallel projects and customers other organisational units, know and accept the requirements

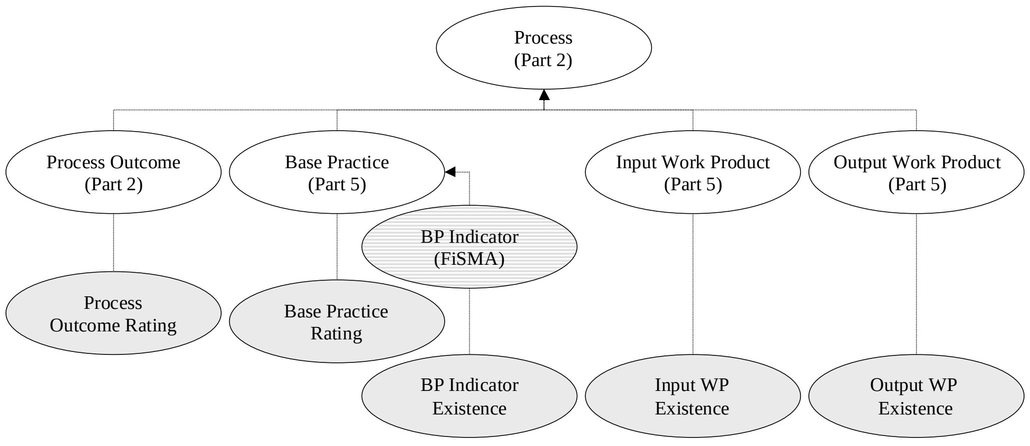


Fig. TMAKINEN.2 : The model of assessment indicators on level 1

The assessment indicators for levels 2 to 5 are shown in fig. TMAKINEN.3. Capability rating of levels 2 to 5 can be done by rating the process attributes defined in SPICE Part 2. The more detailed SPICE Part 5 presents management practices (MP), which can also be rated. The indicators of the management practices are generic, i.e. the same for all processes, and the interpretation is the responsibility of the assessor. A remarkable feature of the FiSMA extension is the process specific management practice indicators, which already contain one interpretation. SPICE Part 5 divides the management practice indicators into practice performance characteristics, resources and infrastructure characteristics and associated processes. The FiSMA model also includes associated work products for the management practices.

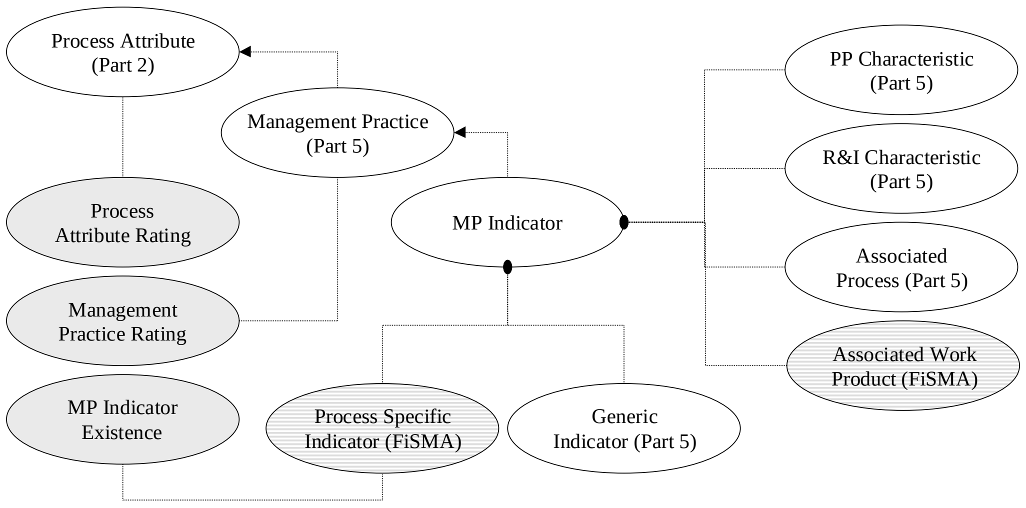

Process Attribute (Part 2)
Management Practice (Part 5)
Process Attribute Rating
MP Indicator
Management Practice Rating
MP Indicator Existence
Process Specific Indicator (FiSMA)
Generic Indicator (Part 5)
PP Characteristic (Part 5)
R&I Characteristic (Part 5)
Associated Process (Part 5)
Associated Work Product (FiSMA)

Fig. TMAKINEN.3 : The model of assessment indicators on levels 2 to 5

4. The assessment process

The assessments are conducted according to a process utilising the previously described model. The assessment process and its work products are tailored according to the needs of small companies. The five steps of the assessment process and the related work products and tools are presented in fig. TMAKINEN.4. The method is compatible with SPICE and has been tested in several assessments. The goal of the process is to provide detailed improvement ideas and actions while being light enough for the smallest software organisations. The assessors can benefit from tools that support the assessment. The tools can be e.g. forms, checklists and templates for both carrying out the assessment as well as recording the results for further analysis.

The initiative for the assessment is typically a result of the SPI priority setting. A preliminary SPI plan defines the area of the SPI work and guides the selection of the processes for the assessment.

The selected processes are assessed in detail using the SPICE Part 5 compatible method. Assessments are carried out using the assessment forms distributed by FiSMA. The assessors can be either external or internal (i.e. self-assessment) to the company.

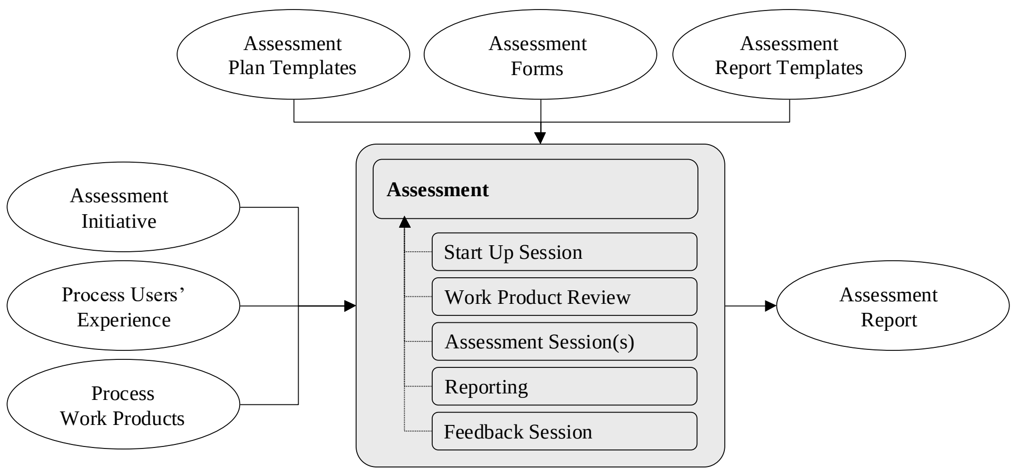


Fig. TMAKINEN.4 : The assessment process

The first two steps of the assessment are the start-up session and the work product review (fig. TMAKINEN.5). In the start-up session the team to be assessed gets information about the assessment, the goal and processes are revised and the assessment schedule is closed. The background information of the assessment instances is collected and documented in the assessment plan. Preliminary material is collected for the work product review. Additional material can be reviewed during the assessment. A written agreement describes the collected material and the handling of the material to ensure confidentiality. The size of the assessment and the experience of the team to be assessed affect the time needed for the start-up session but it typically lasts 1 to 3 hours. The assessors need adequate time for the work product review and the assessment session should commence one week after the start-up session, at the earliest.

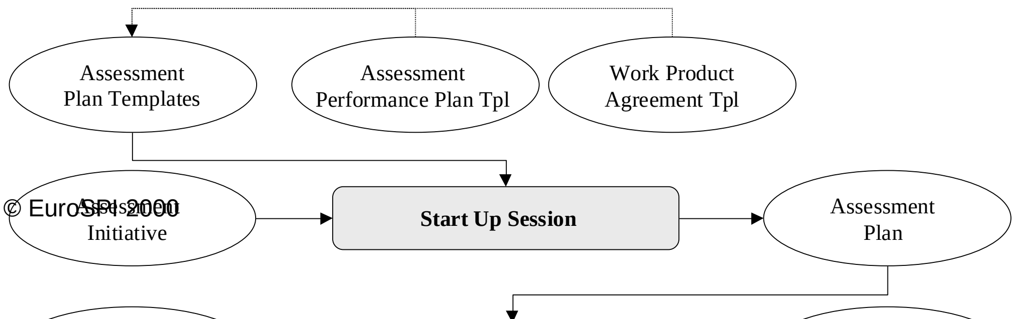

Assessment Plan Templates
Assessment Performance Plan Tpl
Work Product Agreement Tpl
Assessment Initiative
Start Up Session
Assessment Plan

Fig. TMAKINEN.5 : The assessment process, steps 1 and 2

The third step of the assessment is the most interactive part of the work. The assessment session (fig. TMAKINEN.6) is a discussion between the assessors and the team to be assessed. An assessment session for each process takes on an average two hours. The assessors fill in the assessment forms, based on the discussions with the project team. Since the work products are often referenced during the discussion, sufficient work product review notes help the assessors to find the evidence for the rating. The key results are the initial indicator ratings and detailed notes of the discussions including the improvement ideas. The detailed output of this step improves the accuracy of the assessment.

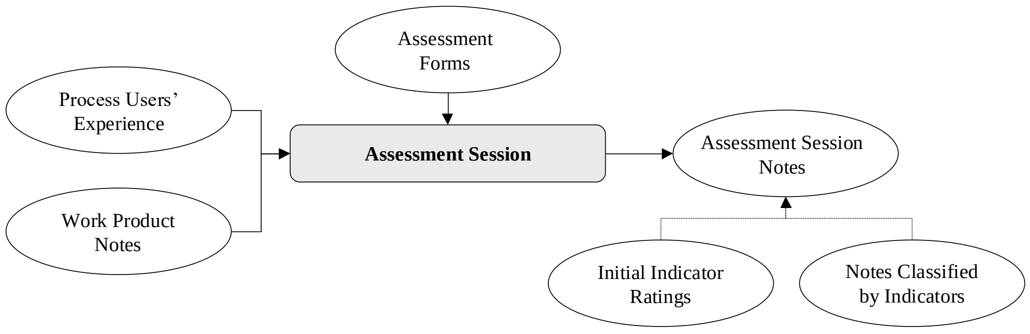

Assessment Forms
Process Users' Experience
Assessment Session
Assessment Session Notes
Work Product Notes
Initial Indicator Ratings
Notes Classified by Indicators

Fig. TMAKINEN.6 : The assessment process, step 3

The feedback session (fig. TMAKINEN.7) is held after the assessment report is finished, usually within two weeks after the assessment. The assessors write the assessment report, which contains process ratings, improvement ideas and recommendations for the organisation as a basis for improvement action planning. The assessment session discussions, detailed ratings and existence checks are documented in the report annex. The aim is that the report is written in such detail that it is comprehensible even after several months.

The time needed to complete the report is usually close to the total time needed for the assessment sessions. The session typically lasts two hours, and participation of the management and the assessed team is advisable. The final version of the report can be released after the feedback, which enables corrections and modifications of the report to promote its later use. The feedback session is usually followed by a discussion about the SPI programme preparation start-up.

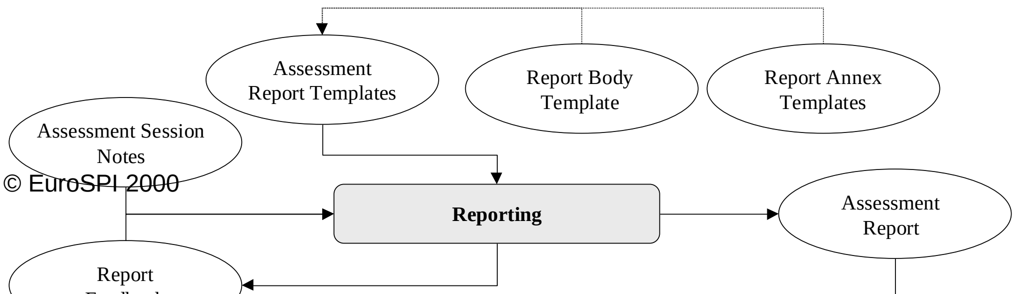


Fig. TMAKINEN.7 : The assessment process, steps 4 and 5

Tools to support the assessment process include templates and forms (fig. TMAKINEN.4). The assessment plan templates consist of two documents (fig. TMAKINEN.5). A performance plan is used to describe the assessment instance, possible restrictions and the schedule. A work product agreement is used to document the preliminary material and its handling.

The assessment forms are based on the model presented in the previous chapter. One form is for level 1 assessment containing process outcomes, base practices and work product lists (fig. TMAKINEN.2). There is a second form for the assessment of higher levels, but still process specific, because of the process specific management practice indicators (fig. TMAKINEN.3).

The assessment report consists of two template-supported parts: the report body and the report annex. The annex template is process specific containing the same indicators as the assessment forms. The content of the report is described in the next chapter.

5. The assessment report

The assessment report consists of five chapters and an annex (fig. TMAKINEN.8). The report is compiled so that there are three levels of details (chapters 1 and 3, and the annex) to suit best the intended use of the report. The main purpose of the report is to serve as an input to the SPI programme planning.

The title page of the report identifies the assessment: organisation, report date, assessors and version history. The first chapter, Introduction and Summary of the Results, is the management summary. Background, sponsor, participants and the goal of the assessment are mentioned. The summary part includes the ratings on the process level, the key observations and the improvement ideas. The language should be easy to read even to those without prior experience of assessments or SPI.

Chapter 2, Assessment Goal and Process Assessment Plan, documents the assessment plan and the schedule. Chapter 3, Capability Results of Assessed Processes, describes each process in more detail. The emphasis is on the assessment observations and the improvement ideas. This level of detail is best for the SPI programme planning and the setting of the priorities.

The fourth chapter, Distribution of Practice Ratings, is the statistical part of the report. The assessors present ratings in tables with additional comments. Chapter 5, Guidelines for SPI Planning, contains the assessors' opinion and recommendation to the organisation, how to proceed with the assessed processes.

The annex, Detailed Assessment Results, is the most circumstantial documentation of the assessment. The ratings are presented on the practise indicator level as well as on the practice and the attribute level. The assessment discussions are documented according to the indicators. The annex presents the candidates for improvement actions. The candidates are derived from both the improvement ideas that arose in discussions and from the low-rated indicators. This level of detail is best suited when planning process specific SPI actions. A typical size of the report is 20 to 35 pages.

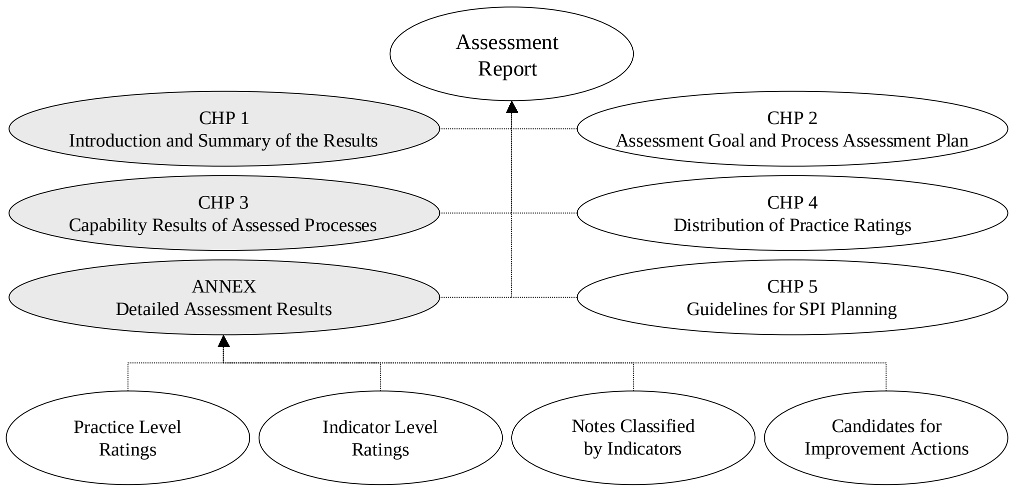

Assessment Report
CHP 1 Introduction and Summary of the Results
CHP 2 Assessment Goal and Process Assessment Plan
CHP 3 Capability Results of Assessed Processes
CHP 4 Distribution of Practice Ratings
ANNEX Detailed Assessment Results
CHP 5 Guidelines for SPI Planning
Practice Level Ratings
Indicator Level Ratings
Notes Classified by Indicators
Candidates for Improvement Actions

Fig. TMAKINEN.8 : Assessment report content

6. Lessons learned

The presented Detailed Process Assessment Method For Software SMEs has been developed and used during the SataSPIN project. In two years the project has performed 11 assessments; summary of the assessments is in the following table (fig. TMAKINEN.9).

Companies assessed		*8*
Projects		*19*
Persons interviewed		*51*
Processes		*39*
Process instances		*58*
Level 1	*Base practices*	*265*
	Instances	*412*
Level 2-5	*Mgmt practices*	*261*
	Instances	*371*
Pages in assessment reports		*339*

Fig. TMAKINEN.9 : Summary of assessments

The assessed companies consider that the assessments are of high importance as a part of their SPI programmes. The companies are also committed to continue the assessments in future years. In addition, some companies have either already used or are prepared to use the method in self-assessments.

The goal of the presented work is to provide a method that is easy to use but yields detailed improvement actions. The beneficiaries of this detailed and tailored assessment model are software companies and assessors. So far our experiences of the tailored assessment method for the SMEs are good: all the assessed organisations are continuing their SPI effort. Nevertheless, the method needs further development of both the model and the process.

6.1 Benefits

The small software companies are the largest stakeholders in the development of this assessment method. Software process assessment requires devotion together with investments from the company being assessed. Because of the scarcity of resources the majority of small software companies would like to have immediate payback from their investment. A longer-term goal is to be able to practice self-assessment – assessment by either one or a group of the company's employees, rather than independent assessment led by an external assessor. In order to carry out self-assessment the assessment models should be as simple as possible, offering guidelines, procedures and detailed checklists, as does the tailored FiSMA model.

The assessors can benefit from detailed models both in quality and efficiency. SPICE process assessment ratings are highly dependent on the qualified assessor’s judgement. The skill and competence of the assessor are important factors, but the basis for repeatability across assessments must be provided by more objective mechanisms [1]. For these purposes there should be forms, checklists, templates for both carrying out the assessment as well as recording the results for further analysis and development of the improvement plans. Based on the detailed ratings and notes the accuracy of the proposed SPI actions improves. Appropriate tools ensure the consistency of the process and save time in the reporting.

6.2 Further development

The assessment model requires development of the work product characteristics. At present the model contains only the existing work products, but more descriptive work product characteristics could assist in reliable content based rating of the work products. The work products could also be directed to practises and various checklists could be developed to support the assessment steps. The idea of assessing the management practises in connection with each base practice is also under evaluation.

Support for continuous SPI is a natural development goal for the process. The next phase of the SataSPIN project will support the assessed companies with consultation and company specific assistance in further assessment activities enabling continuous process improvement. One of the supportive activities should be training the companies to become competent in self-assessment. There is also a need for an outside consultant once in a while to encourage the employees to update the improvement plans or to help in brainstorming events to find proper improvement actions. Having seen the first SPI benefits the small organisations also start to believe in the usefulness of assessments in process improvement, and their readiness for continuous SPI increases.

7. Acknowledgements

This work is largely based on the FiSMA assessment method originally developed by Mr. Risto Nevalainen of STTF Oy. Mr. Nevalainen has splendidly promoted the use of SPICE in Finland. The method and the related assessment forms are distributed by the Finnish Software Metrics Association (FiSMA). Today FiSMA links together the SPICE users in Finland and the international development organisations in the field of process improvement.

The SataSPIN companies contributed to the work partly by participating in the development of the methodology and partly by testing the methods in practice. The feedback from the companies has been essential.

The process descriptions in this paper use the graphical notation presented by The Software Productivity Consortium [5].

8. References

[1] Coletta Antonio: Process Assessment Using SPICE: The Assessment Activities, in El Emam K., Drouin J.-N., & Melo W. (eds.): SPICE: The Theory and Practice of Software Process Improvement and Capability Determination, *IEEE Computer Society Press*, 1997.

[2] ISO/IEC TR 15504-2:1998 Information technology - Software process assessment - Part 2: A reference model for processes and process capability.

[3] ISO/IEC TR 15504-5:1998 Information technology - Software process assessment - Part 5: An assessment model and indicator guidance.

[4] Mäkinen Timo, Varkoi Timo, & Jaakkola Hannu: Database Implementation for a Software Process Assessment Model, *Proceedings of the PICMET´99*, Portland, Oregon, 1999.

[5] Software Productivity Consortium: Improving The Software Process Through Process Definition And Modelling, *International Thomson Computer Press*, 1996.

[6] Varkoi Timo, & Mäkinen Timo: Software Process Improvement Initiation In Small Organisations, *Proceedings of the FESMA-AEMES 2000*, Madrid, 2000 (in press).

[7] Varkoi Timo, & Mäkinen Timo: Software Process Improvement Network in the Satakunta Region - SataSPIN, *Proceedings of the EuroSPI´99*, Pori, 1999.

[8] Varkoi Timo, Mäkinen Timo, & Jaakkola Hannu: Process Improvement Priorities in Small Software Companies, *Proceedings of the PICMET´99*, Portland, Oregon, 1999.

Rapid Assessment to solicit Process Improvement in SMEs

Giacomo Bucci
University of Florence, Italy

Maurizio Campanai
Giovanni A. Cignoni
Cesvit SpA, Florence, Italy

In recent years, Software Process Improvement (SPI) has achieved a good level of penetration in medium-large enterprises. Several well-known methodologies exist and are applied as standard tools in the organisational processes devoted to continuous improvement. Unfortunately, this is not the case for many small and micro enterprises that often are simply unaware of the existence of such methodologies.
The need for making such enterprises aware of SPI concepts as well as the potential benefits of their application was one of the driving motivations for the TOPS project, funded by the European Community as the ESPINODE for Central Italy within the ESSI programme.
In order to overcome the typical inertia showed by enterprises toward usual dissemination initiatives, we adopted *rapid-assessment* meetings to allow enterprises to “taste” SPI. Such activity was part of the goals of TOPS toward the regional industry. Rapid assessments, based on the tailoring of SPICE model, have been offered as a free service to enterprises that joined TOPS.
This paper describes rapid assessment as a powerful way for both diffusing software best practices and proposing actual improvement paths to enterprises. Data collected through the assessments are presented together with some findings about software process maturity in Central Italy.

Introduction

Several authors [1, 2] have pointed out that small enterprises find it difficult to adopt the most acknowledged Software Process Assessment and/or Improvement methodologies, such as CMM, SPICE, BOOTSTRAP, etc., since they are felt cumbersome, overdimensioned and too expensive. In fact, it is hard for a small organisation to relate the scope of consolidated software assessment and improvement techniques to its business objectives.
This is no surprise if one considers how software process is carried out in small enterprises: the software team is usually very small, made of few individuals, with little o no differentiation of roles. In this context the deployment of procedures appears to have an immediate impact only in terms of overhead and burocracy. But if one considers that Information Technology will be one of the key factors driving progress in the 21st century – and that small enterprises will continue to be the majority of software organisations – it becomes more and more important to devise evaluation criteria and procedures for addressing their needs.
The key factor for approaching software process improvement in small enterprises is to understand that their software process has little structure and that they need to identify only a few aspects on which to focus. Furthermore, given the pressure under which software professionals and programmers tend to work, to be successful, a methodology must show measurable benefits in a short time. A reasonable approach is to take a subset of concepts and methods from the above mentioned methodologies and to adapt them to the specific needs of a given organisation.
However, it is not an easy matter to convince small enterprises to plan on a process improvement experiment. One must provide evidence of the bad and the good things of current operation. The objective of the experiment should be to insulate some specific aspect which can be improved and such that the foreseen benefits can be measurable in the short run. However, this requires that the current software process is analysed and assessed and again we are faced with the need to resort to an assessment procedure as "easy" as possible.
This paper describes a *rapid software assessment* technique which has been defined and successfully applied by *TOPS* (TOPS stands for Toward Organised Processes in SMEs) within the context of the *ESPINODE* initiative [3] funded by the European Commission within the *ESPRIT-ESSI* programme.
The ESPINODE network links a set of 18 projects (17 in Europe plus one in Israel). All nodes aim at diffusing Software Best Practices (SBP) and Software Process Improvement (SPI) concepts and methodologies, and supporting enterprises already involved in *Process Improvement Experiments* (PIEs) funded by the ESPRIT-ESSI programme. TOPS is the ESPINODE for Central Italy [4]. TOPS is run by *CESVIT*, the *Agency for High Technology in Tuscany*, whose mission is to promote technological innovation in order to improve the competitiveness of the regional industry.

Project Background

The definition of a rapid assessment technique has been motivated by objectives of TOPS within the ESPINODE initiative. Like the other ESPINODES, the target audience of TOPS is made by regional industry. The regional industry is made by the local enterprises that are interested in improving their software process. Activities targeted to the regional industry are primarily concerned in to promoting awareness event regarding the benefits of SPI, in many cases using results of PIEs as examples.

The problem with the local industry is that small enterprises are reluctant to perform SPI. Usually they do not have neither the resources nor the culture to embark in software process improvement: we cannot ask a small organisation to change the way they produce software, unless we are able to show that there will be an immediate return on the investment. On the other hand, at least in Italy, you rarely find in SMEs a professional dedicated to software process and software quality. Even those firms that understand the potential benefits of improving their software processes have little interest in performing a long term plan. For the same reason, there is no chance to perform a process assessment unless this can be carried out in a rapid manner.

Since the beginning of the project, we have found that the best way to solicit execution of SPI experiments was to establish direct contacts with the enterprises and to perform a rapid assessment of their software process in order to identify and plan specific actions. To this end awareness and training events have been used as a way to establish the very first contact with the enterprises and to present the opportunity of a rapid software process assessment as a free service. The subsequent assessment programme had the following specific goals:

- to stimulate interest in software process assessment and improvement;
- to contribute to the definition of specific improvement plans;
- to collect data and statistics about software process maturity.

In the search for a methodology to use for the rapid assessments, we had to cope with three other requirements:

- to be compliant with the *Regional Industry Survey* promoted by the ESPINODE network;
- to devise tools, questionnaires and a procedure allowing the performing of the assessment in half a day;
- to stay within project budget.

In search for a good compromise between accurate results and low costs, we decided to develop our own methodology. Existent tools, such as, for instance, *Bootcheck* [5], or *Process Advisor* [6], need a full day or more to perform an assessment, while we want to stay in half a day and have time for explaining and discussing SBP and SPI concepts with the enterprise people.

The Rapid Assessment Methodology

The use of questionnaires as a practical base for assessment is very common. The already cited Bootcheck and Process Advisor, for instance, are questionnaire based assessment tools. The TOPS rapid assessment is made with the aid of a questionnaire too. However, in our opinion, the questionnaire is just a schema to carry out the assessment meeting: the skill of the professionals that actually drive the assessment is the most important thing in the success of the meeting.

In the development of the questionnaire we followed the model defined by the SPICE project. SPICE [7] is a well known model and is the base for definition of the ISO/IEC 15504 foreseen standard for software process assessment methodologies. The standard is still in development, last result of the standardisation process are currently published as ISO/IEC TR 15504:1998 [8]. We decide to use SPICE and not ISO 15504 for the following reasons:

- SPICE documentation is publicly available, ISO documents are not; we use the questionnaire to diffuse SPI and SBPs, then publicly available documentation is an important issue;
- the ISO standard is non yet definitive, it is a TR and updates are expected until the final standard will be released (technically it is a TR type 2, i.e. a document subject to review for the proposal of a standard – note that TR are not standard but just ISO documents and that not all ISO TR became a standard);
- differences between SPICE and ISO 15504 mostly concern details and terminology without changing the basic SPICE concepts;
- after SPI and SBPs diffusion, the goal of our assessment is to find improvement paths; because capability certification is not the goal, there is no need of a reference standard.

We also decided to refer the first SPICE model, that identify 35 processes, because, for our purposes, we feel it more accurate than the second SPICE model (29 processes). Yet, with a limited effort, it will be possible to map the questionnaire results in the other models, among which the SBP Questionnaire [9] developed within the ESPRIT-ESSI programme. In this perspective, results of our assessment can be directly used in the survey promoted by the European Commission.

Being "rapid", the methodology is necessarily approximate. Due to time constraints we sacrifice the scope and the accuracy of the assessment, since the assessment meeting has to last only half a day, including time for discussion. In particular, a very general assessment is made on the 35 processes and some more accurate questions are made about just three processes belonging to two of the five SPICE process categories. Moreover, accuracy of the assessment is limited to the answers given by the enterprises; in other words, we do not look for sources of evidence or probe for process management indicators.

The Questionnaire

The methodology is based on a three part questionnaire. All parts are to be compiled by TOPS professionals with a substantial experience in software process assessment, quality management and software engineering related activities. However, data relevant to Part 1 (QP1) are collected by phone, while Part 2 and 3 (QP2, and QP3 respectively) are compiled within a direct audit meeting. The Questionnaire is public and available on the TOPS Web site [10].

The aim of this QP1 is to collect general information concerning the enterprise, including enterprise characteristics such as company type, dimensions, turnover, and so on. It also aims at identify enterprise goals for the near future, awareness about SPI methodologies and benefits, as well as knowledge of European initiatives to support enterprises in their SPI experiments. Answers are predefined (excluding of course those that refer to enterprise data) and the enterprise has to pick the most applicable one.

QP2 is organised in three sections respectively aimed at: *i.* collecting some general data as regards the software development unit, such as dimensions, type of life-cycle, type of software products, and so on; *ii.* assessing the organisational and technological level of the software development unit; *iii.* assessing software process awareness in the enterprise.
The goal of QP2.*ii* is to evaluate how much the enterprise is well organised and technologically mature and thus some questions regard issues that are considered Software Best Practices (SBP). In the analysis of the assessment results, we do not use data from QP2.*ii* to define the software process maturity; instead we use them to give an evaluation of the improvement capability of the enterprise. A well defined organisation is a strong advantage in the enactment of a software process improvement action. From another point of view, technology is often a factor of change: for instance when the introduction of a development methodology pushes forward the formalisation of the development process.
QP2.*iii* aims to evaluate how much the enterprise is aware of the many components of the software process. With respect to the 35 key processes identified by the SPICE model, enterprise has to declare the status of each process, where a process can be:

- *known*, if the enterprise is able to describe the process in terms of goals and activities and recognise it as a component of its business processes;
- *performed*, if the enterprise actually performs, either in a formal or informal way, the process as one of its usual processes, so that, for instance, there are resources allocated to the process tasks and there is evidence of the process performance;
- *defined*, if there exists procedures that define how the process has to be performed, for instance addressing planning and documentation of the process;
- *critical*, if the process is perceived as critical, where the meaning of critical is the most large one: critical because of costs, because it is not well applied, because it is really an important process for the enterprise business perspective, and so on.

Note that if a process is not known, it cannot be declared performed, nor defined or critical. To limit the number of critical processes, enterprise can select a maximum of 2 critical process for each category and a total maximum of 8 critical processes.
QP3 finally is the true software process assessment that is made with respect to three specific processes of the SPICE model: *ENG.2 develop software requirements*, *ENG.5 integrate and test software*, and *CUS.4 perform joint audits and reviews*.
We restricted the assessment to these processes mainly for time constraints: we want to have time for discussion rather than to submit the enterprise to a long succession of questions that need quick answers. We also want to avoid the risk of “difficult” topics, as for instance configuration management, that may need explanation diverting the focus of the meeting. Moreover, the chosen processes, because they directly impact product quality and customer satisfaction, are often naturally seen as the ones in which improvement is more important and then they are the most suitable for small improvement projects. In this sense, we privileged the use of the assessment meeting as a means to pick out improvement opportunities and suggest a consequent experiment. Results of QP2.*iii* demonstrated that the choice was appropriate since enterprises perceive ENG.2 and ENG.5 as the most critical processes.
As shown in the example reported in Fig. BCC.1, for each specific process there are five questions, each one corresponds to a maturity level and summarises the best practices of that level – i.e. the generic practices of the SPICE model. Answers to many of the questions of QP2 and QP3 are of yes/no kind. To answer *yes*, there must be an appropriate documentation proving that processes actually follow the best practice presupposed by the question. To the yes/no answer a percentage value is added. In the *yes* case, the percentage represents how much the activity is performed as stated; in the *no* case, percentage represents how much the practice is not stated but de facto applied, for instance by customer request or personal initiative. While *yes 100%* is the best situation and *no 0%* the worst, *yes 0%* (a practice stated but completely disregarded) and *no 100%* (a praxis established but not formalised) are still possible answers.

We use the percentage value to have a measure of the distance between the actual situation and the ideal one. The measure can be interpreted in terms of needs of internal diffusion (*yes* with low percentages) or needs of procedure formalisation (*no* with high percentages).

Integrate and test software (ENG.5)

The purpose of this process is to integrate the software unit and assure that the software satisfy the requirements. This process is accomplished through developing aggregates of software units and testing them as an aggregate, and then testing the resulting integrated software. The ENG.5 process is defined in terms of the following activities:

ENG.5.1	determine regression test strategy,
ENG.5.2	build aggregates of software units,
ENG.5.3	develop tests for aggregates, test software aggregates,
ENG.5.4	develop tests for software,
ENG.5.5	test integrated software.

a.	Do software integration and tests produce specific plan and reports?	yes/no	__%
b.	Are software integration and tests defined and planned before the beginning of the coding activities?	yes/no	__%
c.	Is there a standard procedure for planning and executing software integration and tests and reporting their results?	yes/no	__%
d.	Is there a standard procedure to measure and control test effectiveness with respect to predefined goals?	yes/no	__%
e.	Is there a standard procedure for the study of past test results aimed to prevent future software defects?	yes/no	__%

Fig. BCC.1: An excerpt of the TOPS questionnaire, questions about ENG.5

Final results of the assessment, a part of the maturity results in the three assessed processes, are expressed in terms of *actual maturity level* and *potential maturity level*. Actual maturity level is defined as in the SPICE model and requires full application of stated practices. Potential maturity, instead, captures the situations in which maturity level is not reached because of lack in procedure application or because it exists as a praxis that demonstrates the awareness of the best practice. Organisational and technological level are also calculated as an output of the QP2.*ii* results and expressed as a score in the range 0-5. They are computed for the purposes of evaluating the enterprise improvement capability and by no means they are related to the software process maturity level.

Performing a Rapid Assessment

We think that process assessment based on a self-compiled questionnaire may lead to wrong results. When the questionnaire regards process quality, i.e. the quality of the way in which people are used to work, the self-compilation often results in a lack of realism reflecting a "good" or "bad" vision closely related to the feeling of the compiler. For instance, a workshop on SPI and SBP, while can be the way to perform many assessments at one time, it is not a solution to improve accuracy. The event, instead of being a guide to questionnaire compilation, works as an amplifier of feelings, resulting in "very bad" or "very good" visions. Furthermore, with self-compilation we lose the opportunity to establish a true contact with the enterprise.

Then, we decided that TOPS skilled professionals should compile the questionnaire by interviewing the enterprises, explaining the questions and discussing the answers. In spite of its higher cost, this technique is more accurate. Moreover it is less tedious and more useful for the enterprise, which has the opportunity to meet expert people without any direct cost.

We use awareness and training events organised within TOPS as a mean to contact regional industries potentially interested in the target enterprise role. Even if event participation is for free, investing some person/time to participate is an indicator that the enterprise is potentially interested in a future SPI experiment.

Then we contact by phone the enterprise for QP1 compilation. QP1 is an occasion to talk about SPI and SBP and serves also to enterprise selection for the assessment meeting. TOPS funds are limited and then we want to accurately aim our effort towards enterprises. We do not want to bother enterprises with initiatives they are not interested in: the assessment has to be an effective opportunity for their business.

Data from our Survey

During the project, from April 1998 to May 2000, TOPS collected 95 QP1 to be used for the Regional Industry Survey and 36 rapid assessments based on QP2 and QP3. Here we present some general considerations about the results of our survey. A previous analysis of a partial set of assessment was in [11]. Full aggregate data and analysis are available as project technical reports on the TOPS Web site.

The Regional Industry Survey

The large majority of enterprises are very small both in terms of employed people and turnover. This reflects the typical situation of central Italy enterprises and, in fact, TOPS has SMEs as a specific target audience. The geographical distribution of the collected questionnaires reflects the localisation of CESVIT (in Florence) and Innova (partner of TOPS in Rome), with the majority of the interviewed enterprises located in Tuscany, Latium and Liguria.

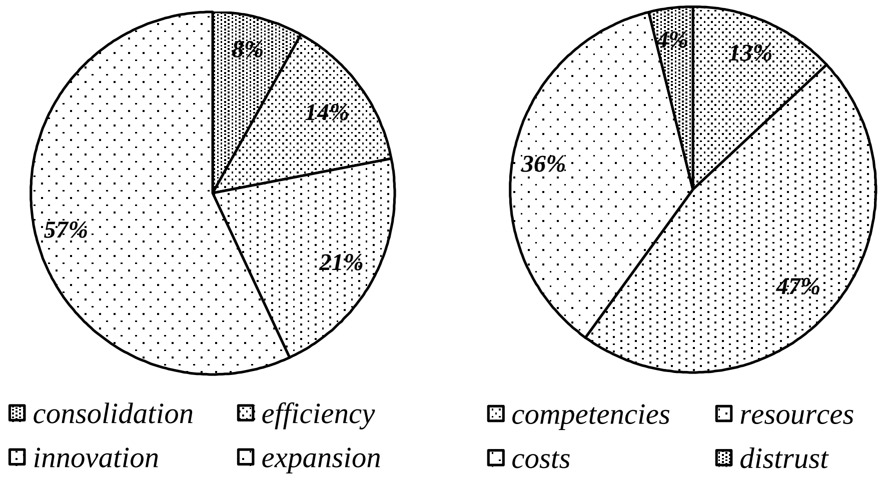


Fig. BCC.2: Enterprise goals and SPI barriers

It is remarkable that the large majority of the interviewed enterprises has expansion (57%) or innovation (21%) as their goals for the near future. On the other part, very few of them want to make their processes more efficient (14%) or consolidate their market position (8%). This depicts a situation in which SPI can rely on very few resources: all enterprise efforts are directed to new targets and not towards consolidation and improvement of the already reached ones. As a natural consequence major barriers to SPI are costs and resources (83%); noticeably very few enterprises do not invest in SPI because of distrust in its benefits (4%). When and where some SPI initiative is performed (50%), the most pursued goal is efficiency (50%).
In conclusion, enterprises have lot of business opportunities and logically want to exploit them as much as possible. Quality is put in background and SPI is principally viewed as a way to be more profitable. These consideration are even more sharp if we consider very small enterprises (under 50 people): expansion is the main goal for the 62% and only 43% perform SPI. Fig. BCC.2 shows data about enterprise goals for the near future and general barriers to investments in SPI activities.

Software Process Awareness

QP2 and QP3 were submitted to a subset of the enterprises that received QP1. This subset has the same general characteristics of the bigger one. From QP2.i we had some more information about the assessed development units:

- 40% of them rely on internal or local customers, a situation that presuppose an elevate grade of confidence often based on personal relations;
- 56% of enterprises does not have at all a quality system and another 23% had only approached its definition.

QP2.*ii* assesses organisational and technological level. Generally enterprises obtain better evaluations in organisational than in technological aspects (average organisational level is 2.59 where average technological level is 1.75). Going to the details, poor technology evaluations mainly depend by the scarce adoption of tools for project management, version and configuration control, test automation and software measurement. In many cases, regarding pure product technology, enterprises are strongly committed in innovation, for instance about RAD and CASE tools (72%). So we have to read the result not as poor technology at all, but as poor technology in support of process and quality control.
QP.*iii* aims to evaluate how much the enterprise is aware of the many activities of the software process. During the assessment meeting we ask the enterprise to briefly describe how they perform each one of the 35 SPICE processes and, on the basis of their answers we classify the key-process as *known*, *performed* or *defined*. On the average, enterprises declare that, out of 35 SPICE, processes 32 are known, 25 performed and 14 defined. However, despite these apparently good results, we had the feeling that many enterprises were reasoning about their process for the first time.
Using these data it is possible to have a first approximation of the maturity level. Setting *performed* as correspondent to level 1 and *defined* as correspondent to level 3, we can calculate an average maturity level of 1.4. In fact, as we will show later, this is an optimistic result.
More interesting are data about processes that enterprises perceive as critical. The most critical processes are ENG.2 and ENG.5: two of the three processes we selected as the ones subjected to the “detailed” part of our assessment (QP3). Some oddities reveal the naive approach of many enterprises. For instance, *PRO.6 Manage risks* is another very critical process that also results to be one of the less known, and it is perceived more critical than, for instance, PRO.4 Manage requirement and PRO.5 Manage quality. As a possible interpretation, many of the assessed enterprises suffer risks because do not manage requirements nor quality. As a consequence, they feel critical the process that they are compelled to perform. They instead forget the processes that they should perform and that are the actually critical ones.
We consider this part of the assessment very important because it often results in discussions about goals and responsibilities of the various processes. With respect to the “awareness” goal of the initiative, this part of the assessment is a good mean to show to enterprises the many facets of the software process and to highlight the difference between a naive and a structured approach to software development.

Result from the Rapid Assessments

Figures BCC.3 and BCC.4 show the results of the rapid assessments. Data are presented as the average on the number of enterprises of the maturity levels calculated as defined in the SPICE model.
Results about actual maturity are very poor. The very large majority (84%) of enterprises scores a 0 level: they have no documentation evidence that activities are performed. Average value is 0.3 and the maximum scored level is 3.
Potential maturity scores, while not exciting, are better, with an average value of 1.8 and a maximum score of 3.7. This reflects a usual situation: there is a general knowledge of SBP but, instead as being received as enterprise standards, they are applied by demand, for instance in the most critical projects.
To tune our assessment tool, 6 of the 36 enterprises we assessed were selected between enterprises that had a certified quality system and already had some rigorous SPI experience, for instance in an ESSI-PIE or in a SPIRE experiment [12]. Some of them already received a formal assessment using CMM or Bootstrap or participating in the SPICE trials. In these cases

our assessment showed an average error of ± 0.18 in terms of potential maturity level.

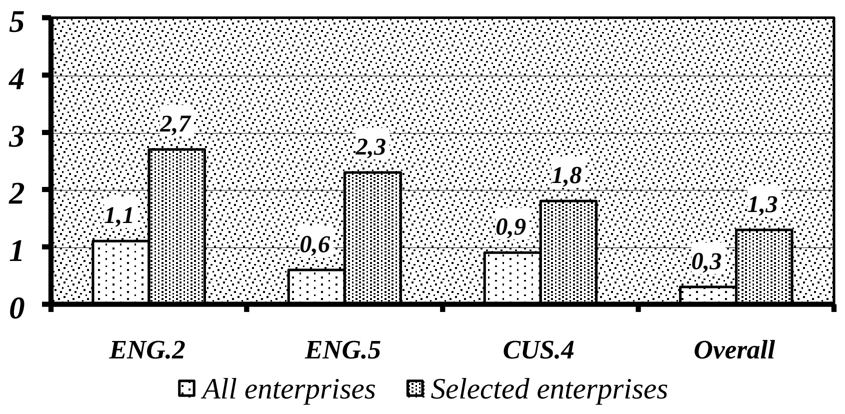


Fig. BCC.3: average actual maturity levels

Because the maturity scores of the selected enterprises are far better than the average, we use these results to stimulate the regional industry: while the average is low there are some excellent performers that demonstrate that SPI is a valuable investment.

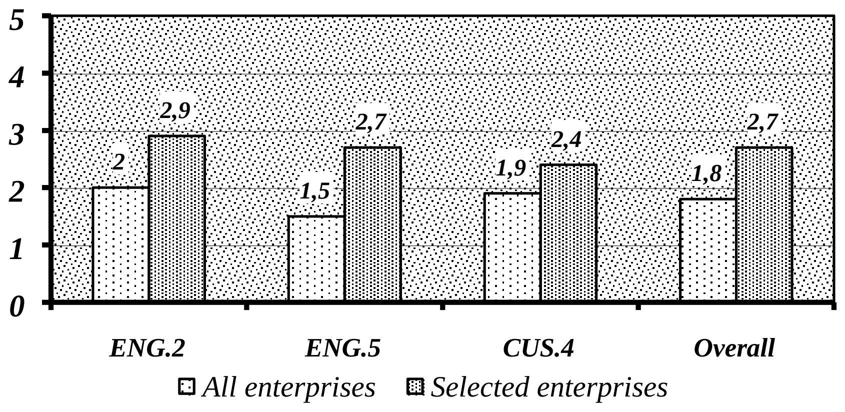


Fig. BCC.4: average potential maturity levels

Conclusions

Having performed a sufficient number of rapid assessment, we can draw the following conclusions. These conclusion refer to the situation in Central Italy.

- Small enterprises (whose customers usually belong to the same regional area) feel not to have a strong need for process quality. However, since they are committed to grow, there is a potential for more organised processes.
- The survey shows enterprises that have grown as their main goal, but score low maturity levels and display very scarce intention about SPI investments. It is still a sector in which enterprises want to be subsidised for improving their organisation.
  Maturity level is low, however it is possible to argue that SBP are in general known, though not applied. In many cases documentation of procedures and internal training are viable paths to achieve higher maturity levels.
- Technologies to support process control are not applied.

The rapid assessment procedure, offered through awareness and training events, allowed us to show that in very many cases, identifiable benefits could be achieved via focused SPI projects.
The rapid assessment initiative is a clear success: 8 of the 36 target enterprises we assessed asked specific advising services to enact improvement activities identified during the assessment meeting. Six of them accepted our services, in some cases partially funded by local initiatives to promote innovation. In one case our advice service lead to a proposal for a Best Practice project funded by EC. In conclusion, for a significative number of target enterprises, our rapid assessment was the starting event of an improvement process. We regard to this as an important achievement.

Acknowledgements

We thank all the enterprises that participated in TOPS activities, without them this work had not been possible. We also thank Francesco Capanni (CESVIT) and Giulio Ravizza (Innova) that helped in the performing of the many interviews and assessments.

References

[1] S.L. Pflegeer et al, “Status Report on Software Measurement”, IEEE software March/April 1977, pp.33-43

[2] K. Kautz, “Making Sense of Measurement for Small Organizations”, IEEE software March/April 1999, pp.14-20.

[3] European Systems and Software Initiative, “ESPINODE: A new scheme to implement take-up measures”, Internet document available at http://www.cordis.lu/esprit/src/enodegen.htm, September 1998.

[4] TOPS Project, “TOPS Terms of Reference”, TOPS Technical Report TOR, available at http://www.cesvit.it/tops/docs/TOPS-tor.pdf, June 1998.

[5] EUROPEAN SOFTWARE INSTITUTE, “BootCheck Overview”, Internet document available at http://www.esi.es/Bootcheck/, October 1998.

[6] PRESSMAN, R.S., “Process Advisor”, RSP&A Inc., 1992.

[7] SPICE Document Suite, Internet document available at http://www-sqi.cit.gu.edu.au/spice.

[8] International Organisation For Standardisation, “ISO/IEC TR 15504:1998 Information technology – Software process assessment — Parts 1-9”, 1998.

[9] European Software Institute, “1997 Software Best Practice Questionnaire – Analysis of Results”, ESI Technical Report, available at http://www.esi.es/Publications/Reports/tr-sbpqaor3.html, December 1997.

[10] TOPS Project, “SME&SPI Questionnaire”, Annex of TOPS Technical Report TR5, available at http://www.cesvit.it/tops/processosw/TR5-anx1.htm, February 1999.

[11] G.A. Cignoni, "Rapid software process assessment to promote innovation in SMEs", proc. of the European Software Day at Euromicro 99, Milano, September 8-10, 1999.

[14] The Spire Project Team, The SPIRE Handbook: Better, Faster, Cheaper Software Development in Small Organisations, The European Commission, December 1998.

A New Approach to Benchmark-Based Process Improvement

Yingxu Wang
Dept. of Electrical and Computer Engineering
University of Calgary
2500 University Drive, Calgary, Alberta, Canada T2N 1N4

IVF Centre for Software Engineering
Argongatan 30, S-431 53, Molndal, Gothenburg, Sweden
Yingxu.Wang@acm.org

Graham King
Design and Advanced Technology Research Centre
Southampton Institute
East Park Terrace, Southampton, SO14 0YN, UK
Tel: +44 23 80 319551 Fax: +44 23 80 319722
Graham.King@solent.ac.uk

ABSTRACT: Although the conventional goal-based process assessment and improvement technologies have been widely accepted, its philosophy of “the higher the better” has been questioned in practice. Particularly it is found that the determination of target capability levels for a specific organization tends to be virtual, infeasible, and sometimes overshoot. Benchmark-based process assessment and improvement provides a new approach to adaptive and relative process improvement based on a philosophy of “the smaller the advantage, the better.” According to the benchmark-based process improvement method, the target capability levels of given software processes may be set relative to the benchmarks of the software industry, rather than to the virtually highest capability level as in a goal-based process model.

Benchmark-based process assessment and improvement is a cutting-edge technology in software engineering for adaptive and relative process improvement. A series of surveys have been conducted worldwide in order to establish a set of benchmarks for software engineering processes on the basis of a comprehensive software engineering process reference model known as SEPRM. With the SEPRM benchmarks, the new approach of a benchmark-based process assessment and improvement is enabled.

Key words: Software engineering, software process, model, base process activities, survey, benchmark, benchmark-based assessment/improvement

1. Introduction

Current software engineering process assessment and improvement approaches are mainly model-based. The classical model-based software process technologies developed in recent decades have dominated the approaches to software engineering process assessment and improvement. This paper intends to develop a new approach to benchmark-based process assessment, and describe its advantages and applications.

In empirical software engineering, it is recognized that for software process improvement, an organization may not necessarily to get all of its processes at the highest level to be competitive, because this would not be the best, most feasible and most economical solution for the organization. Instead, the best solution is just to have a marginal competitive in each process than the competitors.

This observation inspires a new approach to software engineering process improvement – the small the advantage, the better. In the light of this philosophy [3], a new method of benchmark-based process improvement is developed in this paper.

2. Benchmark-based software process improvement

A software process benchmark is an average reference value that the process statistically performs in a given sector or a given region. Software process benchmarking is one of the important methodologies in software process engineering. Benchmarking is useful in both process assessment and improvement.

In contrast to the model-based approach, benchmark-based process assessment and improvement are a new approach to process-based software engineering. The philosophies of both model-based and benchmark-based technologies are contrasted in Table 1.

Table 1. Different philosophies in process assessment and improvement

Subject	Model-based SPA/SPI	Benchmark-based SPA/SPI
Philosophy	The higher the better	The small the advantage the better
Assessment results	Absolute process capability levels	Relative process capability gaps
Target process capability levels for improvement	Set to absolute higher levels	Set to marginally higher than benchmarks
Usage	Support for virtual aims in competition	Support for adaptive aims in competition
Advantage	Widely practiced and experienced	A refined methodology
Drawback	Target capability level might overshoot and not feasible	Need to maintain a set of benchmarks

It is obvious that the implementation of a benchmark-based process assessment is dependent on the availability of process benchmarks. To enable this methodology, process benchmarks are required to indicate the current levels of process capabilities in each individual business sectors in the software industry.

There have been few small-scale process benchmarks, such as the European process benchmarks developed by IBM [1] and the Swedish national software engineering process benchmarks [2]. The IBM benchmark on software engineering practices in Europe contains benchmarks for 69 practices in 7 processes. A comparison between the European and the Swedish benchmarks on the same set of software processes is illustrated in Figure 1.

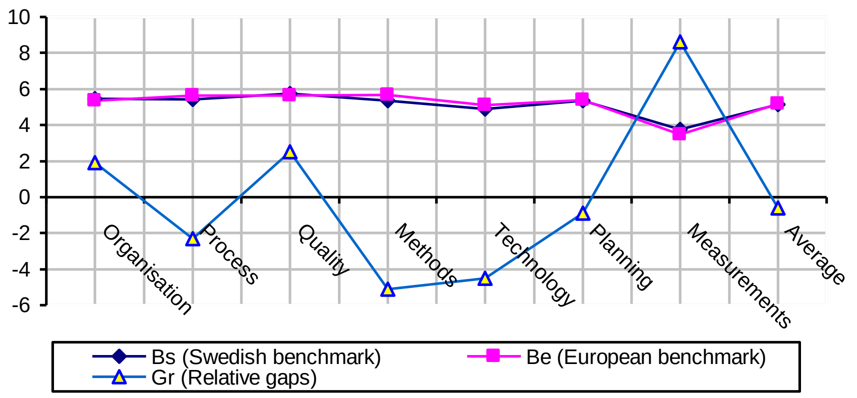


Figure 1. Comparison between the European and a national benchmarks of software engineering processes

It is interesting to find that the Swedish national benchmarks are quite close to the European ones. The average difference of all processes is only –0.6%. This means that the software engineering practices in the Swedish software industry have generally reached the European best practice level.

Magnified gaps between the two benchmarks are shown by the G_r curve. As shown in Figure 1, G_r indicates that, as referred to the European benchmarks, the organization, quality and measurements practices of the Swedish software development organizations have exceed the European benchmark; however the process, methods, technology and planning practices are below the European benchmarks.

3. Establishment of a Benchmark-Based Process Model

The comparative analysis described in Section 2 indicates new possibilities for benchmark-based software process assessment and improvement. However, The IBM and Swedish process benchmarks only covered less than 10 processes. There is thus a need to develop a superset of software benchmarks in order to achieve more accurate results. This section introduces the software engineering reference model (SEPRM) with process benchmarks covering 51 software engineering processes characterized by quantitative process attributes that could be used as the basis for an overall and comprehensive benchmarking system.

3.1 The SEPRM Process Model

SEPRM [3, 4] is developed by integrating major current process models and standards such as CMM, ISO 9001, BOOTSTRAP, ISO 15504 (SPICE), and ISO 12207. SEPRM identifies a superset of software engineering processes that is modeled in 3 process subsystems, 12 process categories, 51 processes, and 444 base process activities (BPAs). A high-level hierarchical structure of the SEPRM framework is shown in Table 2.

Table 2. The SEPRM Process Model

ID.	Subsystem	Category / Process	Identified BPAs
1	Organization		81
1.1		Organization structure processes	13
1.1.1		Organization definition	7
1.1.2		Project organization	6

1.2		Organization processes	26
1.2.1		Organization process definition	15
1.2.2		Organization process improvement	11
1.3		Customer service processes	39
1.3.1		Customer relations	13
1.3.2		Customer support	12
1.3.3		Software/system delivery	11
1.3.4		Service evaluation	6
2	Development		115
2.1		Software engineering methodology processes	23
2.1.1		Software engineering modeling	9
2.1.2		Reuse methodologies	7
2.1.3		Technology innovation	7
2.2		Software development processes	60
2.2.1		Development process definition	12
2.2.2		Requirement analysis	8
2.2.3		Design	9
2.2.4		Coding	8
2.2.5		Module testing	6
2.2.6		Integration and system testing	7
2.2.7		Maintenance	10
2.3		Software engineering infrastructure processes	32
2.3.1		Environment	7
2.3.2		Facilities	15
2.3.3		Development support tools	4
2.3.4		Management support tools	6
3	Management		248
3.1		Software quality assurance (SQA) processes	78
3.1.1		SQA process definition	17
3.1.2		Requirement review	5
3.1.3		Design review	4
3.1.4		Code review	3
3.1.5		Module testing audit	4
3.1.6		Integration and system testing audit	6
3.1.7		Maintenance audit	8
3.1.8		Audit and inspection	6
3.1.9		Peer review	10
3.1.10		Defect control	10
3.1.11		Subcontractor's quality control	5
3.2		Project planning processes	45
3.2.1		Project plan	20
3.2.2		Project estimation	7
3.2.3		Project risk avoidance	11
3.2.4		Project quality plan	7
3.3		Project management processes	55
3.3.1		Process management	8
3.3.2		Process tracking	15
3.3.3		Configuration management	8
3.3.4		Change control	9
3.3.5		Process review	8
3.3.6		Intergroup coordination	7
3.4		Contract and requirement management processes	42
3.4.1		Requirement management	12
3.4.2		Contract management	7
3.4.3		Subcontractor management	14
3.4.4		Purchasing management	9
3.5		Document management processes	17
3.5.1		Documentation	11
3.5.2		Process database/library	6
3.6		Human resource management processes	11
3.6.1		Staff selection and allocation	4
3.6.2		Training	7
Total	3	51	444

3.2 SEPRM Benchmarks of Software Engineering Processes

A set of 51 SEPRM software engineering process benchmarks are derived based on a series of worldwide surveys [3, 5, 6] as shown in Figures 2 – 4, representing the SEPRM organization, development and management process subsystems, respectively.

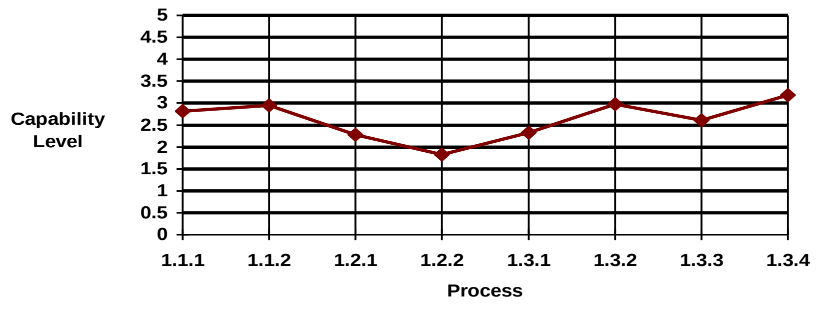


Figure 2. SEPRM benchmarks of organization processes

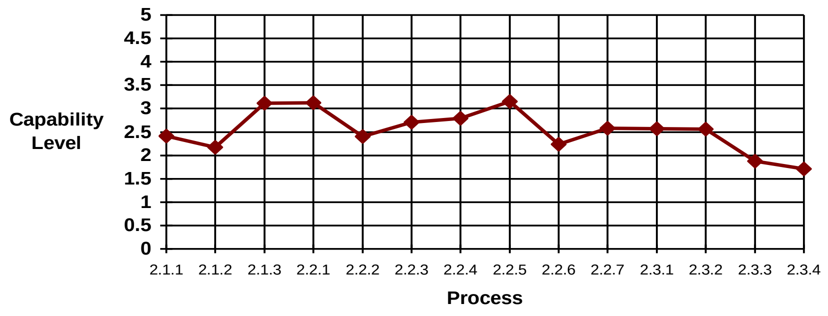


Figure 3. SEPRM benchmarks of development processes

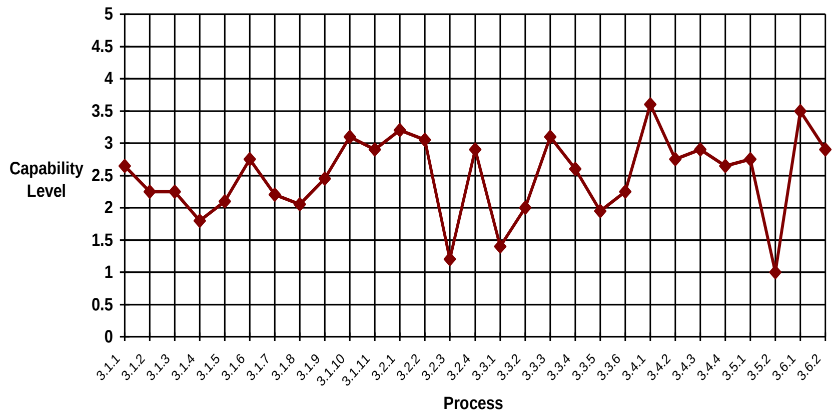


Figure 4. SEPRM benchmarks of management processes

4. Method for Benchmark-Based Process Improvement

When a specific process system profile is obtained in an assessment, and plotted onto the SEPRM benchmark curves, an organization's process capability can be analyzed against the benchmarks. Based on the benchmarks, refined process improvement can be planned in much more accurate manner. This section describes features of benchmark-based process improvement, and demonstrates how a benchmark-based process improvement can be carried out practically.

4.1 Features of Benchmark-Based Process Improvement

The method of benchmark-based process improvement is an extension of the method of benchmark-based process assessment. With regards to the model-based process improvement methodology, the features of a benchmark-based process improvement are as follows:

- The philosophy for a benchmark-based process improvement is to 'filling the gaps' rather than 'the higher the better' as is common in a model-based process improvement;
- The improvement opportunities are identified based on gaps analysis between the plotted process profile and the benchmarks;
- The improvement priorities are determined by quantifying the degree of gaps between the plotted process profile and the benchmarks;
- The improvement achievement is evaluated by checking if the gaps have reduced, and if the process capabilities have been enhanced marginally above the process benchmarks.

4.2 Guidelines for Benchmark-Based Process Improvement

A practical procedure covering benchmark-based process assessment can be carried out in 6 steps as below:

1) Adopt a benchmarked process model
2) Conduct a baseline assessment
3) Plot process capability profile onto the benchmarks
4) Identify gaps between the process profile and the benchmarks
5) Implementing recommended improvement
6) Review process improvement achievements

The following describes how the above processes are conducted in an SEPRM benchmark-based assessment.

4.2.1 Adopt a benchmarked process model

As described in Section 3, the SEPRM is the only benchmarked software engineering process assessment model. SEPRM supports both model-based and benchmark-based software process assessment, and provides a comprehensive set of processes and best practices in the reference model. Therefore, in the following subsections, when we describe a benchmark-based process assessment, we imply that SEPRM and its benchmarks are used in the assessment.

4.2.2 Conduct a baseline assessment

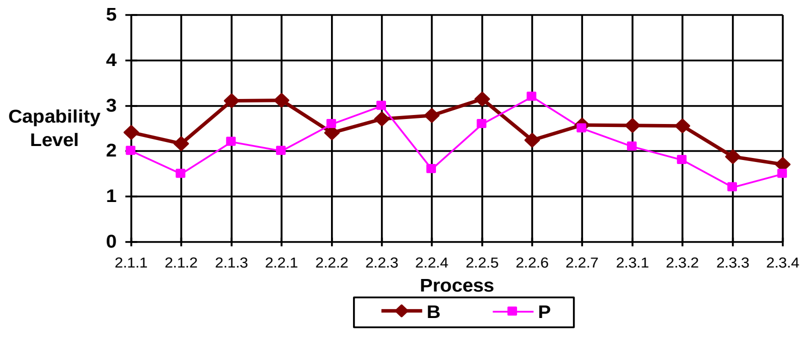
In this step, a process capability profile of the assessed organization will be derived according to the adopted process model. The first assessment is called baseline assessment because it will provide a basis for understanding the capability of existing processes, and for identifying process improvement opportunities.

4.2.3 Plot process capability profile onto the benchmarks

When an organization's process capability profile is baselined, the key differences between the model-based and benchmark-based assessments are on selection of target capability levels and related analysis methods. Model-based assessment usually specifies an absolute higher capability level as the target capability level; while the benchmark-based assessment adopts a set of relative and dynamic target capability levels, the benchmarks, in analysis.

For instance, an organization's assessed capability levels for the 14 SEPRM development processes can be plotted against the benchmarks are as shown in Figure 5. With the support of the SEPRM benchmark curve, analysis of the assessment results is simplified as in the same way as for gap identification.

The organization and management processes modeled in SEPRM can be plotted in the same way, by using the benchmarks provided in Figures 2 and 4.



B – Benchmark curve; P – Assessed process capability curve

Figure 5. Plotted process capability profile against SEPRM benchmarks

4.2.4 Identify gaps between a process profile and the benchmarks

The gaps between a plotted process profile (P) and the benchmarks (B) can be analyzed based on Figure 5. Using the above figure, magnified gaps (G) between the current process capability levels and the benchmarks can be derived as shown in Figure 6. Figure 6 indicates that the larger the gaps below the average, the weaker the process capability is.

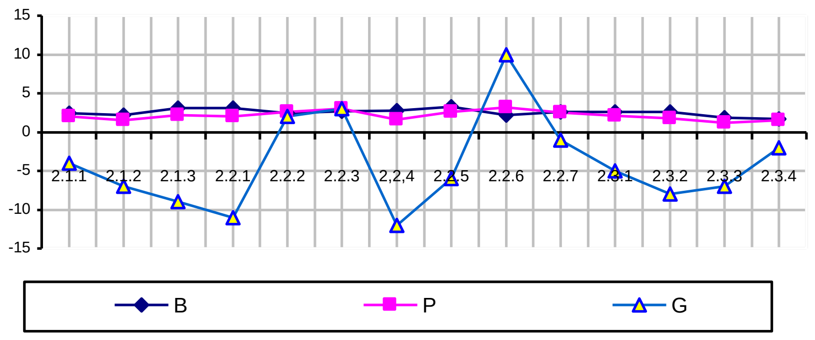


Figure 6. Capability gap analysis in a benchmark-based assessment

1.1 4.2.5 Implementing recommended improvements

With the gaps identified and target process capability levels determined by using the SEPRM benchmarks in the above step, process improvement can be achieved by the following techniques:

- Enhancing inadequate processes
- Replacing ineffective processes

- Introducing new processes
- Reengineering process systems and interfaces

A process improvement program needs to be closely monitored according to an improvement plan in order to ensure tasks progress as expected, implementation is correct, and achievement is realized. If any problems are encountered, causes should be analyzed and the action plan adjusted.

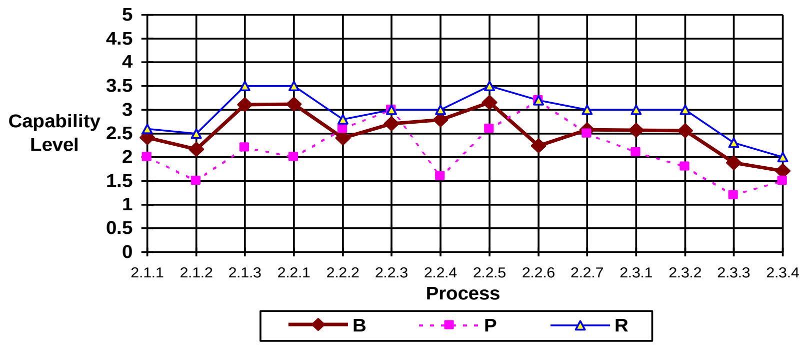
1.2 4.2.6 Reviewing process improvement achievement

The purpose of this step is to confirm whether the planned improvement goals and target capability levels have been achieved. Measurements of process effectiveness may be used to confirm achievement of process effectiveness targets. The possibility of undesirable side-effects should be investigated.

It is a common notion to undertake a new process assessment in order to review improvement achievement. The review assessment can be either a self-assessment or a third-party assessment. Though the review assessment may focus on the affected processes in the improvement program, the whole process system should be assessed in order to find the impacts of improvement on other processes, or any possible side-effects or new imbalance of process capability.

4.3 An Example of Benchmark-Based Improvement

A case for demonstrating an organization’s baseline and improved capability profiles in a benchmark-based process improvement is shown in Figure 7. Figure 7 shows, the baseline process capability profile (P) of an organization has been improved to an adaptive process profile (R) that is marginally above and along with the benchmarked curves (B).



B – Benchmark curve; P – Baseline process profile; R – Improved process profile

Figure 7. SEPRM benchmarks-based process improvement

Note in Figure 7 only the development process benchmarks and profiles are shown. By adopting the SEPRM benchmarks provided in Figures 2 and 4 [3, 5, 6], the improvement of organization and management process subsystems can be carried out in the same way.

5. Conclusions

Benchmark-based process assessment and improvement is a cutting-edge technology in software engineering for adaptive and relative process improvement. This paper has demonstrated that benchmark-based process improvement is more adaptable, feasible, and economical in process-based software engineering. By adopting the benchmark-based process improvement, the target capability levels of a software organization may be set related to the benchmarks of the software industry, rather than the virtually highest capability level as that in a model-based process assessment/improvement approach.

1.3 Acknowledgements

This work has been partially funded by the Swedish National Board for Industry and Technical Development. The authors would like to acknowledge the funding organization and the IBM Consulting Group for their valuable support.

1.4 References

[1] IBM (1997), IBM European Benchmark of Software Development Practices, pp.1-40.

[2] Wang Y., Wickberg, H. and Dorling, A. (1999), Establishment of a National Benchmark of Software Engineering Practices, *Proceedings of 4th IEEE International Software Engineering Standards Symposium (IEEE ISESS'99),* IEEE CS Press, Brazil, May, pp.16-25.

[3] Wang, Y. and King, G. (2000), *Software Engineering Processes: Principles and Applications,* CRC Press, USA, ISBN: 0-8493-2366-5, pp.1-752.

[4] Wang Y., King, G., Doling, A. and Wickberg, H. (1999), A Unified Framework of the Software Engineering Process System Standards and Models, *Proceedings of 4th IEEE International Software Engineering Standards Symposium (IEEE ISESS'99),* IEEE CS Press, Brazil, May, pp.132-141.

[5] Wang Y., King, G., Dorling, A., Ross, M., Staples, G., and Court, I. (1999), A Worldwide Survey on Best Practices Towards Software Engineering Process Excellence, *ASQ Journal of Software Quality Professional,* Vol.2, No.1, December, pp. 34-43.

[6] Wang, Y. et al. (1998), A Worldwide Survey on Software Engineering Process Excellence, *Proceedings of IEEE 20th International Conference on Software Engineering (ICSE'98),* Kyoto, April, IEEE Press, pp.439-442.

About the Authors

Yingxu Wang is Professor of Software Engineering in Dept. of Electrical and Computer Engineering at The University of Calgary, and project manager with the Center for Software Engineering at IVF, Gothenburg, Sweden. He received a PhD in software engineering from the

Nottingham Trent University / Southampton Institute, UK. He is a member of IEEE, ACM, and ISO/IEC JTC1/SC7, and is Chairman of the Computer Chapter of the IEEE Swedish Section. He was a visiting professor at Oxford University. He is the lead author of a recent book on *Software Engineering Processes: Principles and Applications*, and he has published over 100 papers.

Graham A. King is Professor of Computer Systems Engineering at Southampton Institute, UK. He heads a research center which has a strong interest in all aspects of software engineering. He was awarded a PhD in architectures for signal processing by the Nottingham Trent University. He is a co-author of a recent book on *Software Engineering Processes: Principles and Applications*, and has authored three books and nearly 100 academic papers.

Session 10 - Danish Experience in SPI

Session Chair:

Risto Nevalainen,

STTF

RAMSES - Rapid Application Development in Military Software Systems

Peter Gungaard Nielsen
Bruhn Newtech A/S, Herlev Denmark

Svend Skafte
Bruhn Newtech A/S, Herlev Denmark

Abstract

Bruhn NewTech is a small company with 35 employees focused on software development for external customers. We have during the RAMSES project gained useful experience with software process improvement (SPI) and obtained results which documents that there are great benefits in SPI also for small companies.

Our process improvement experiment provided some key lessons learned especially applicable to small software development organisations. Standard process frameworks (e.g. DSDM) must be tailored and detailed to the specific needs of the organisation. SPI management is a difficult discipline, and internal and external communication is very important. SPI does require resources, but the benefits exceed the costs and payback starts immediately.

We have implemented a DSDM (Dynamic Systems Development Method) inspired, RAD (Rapid Application Development) based process involving close interaction with the customer throughout the development process. We developed the process and the necessary document templates to support the project team through a RAD project. We also improved our requirements analysis process and developed requirements specification documents supported by visual elements such as user interface designs in order to facilitate the discussions with the users before starting the programming.

The resulting development process showed significant improvements with increased customer satisfaction, reduction of the often very large workload peaks at delivery milestones by 50%, and elimination of the need for overtime payments through a general reduction of 30% in the number of hours exceeding normal working hours.

Introduction

Bruhn NewTech is a small company with 35 employees located in USA, England and Denmark (headquarter). The major part of the software development is carried out from the Danish office having 10-12 SW Engineers working on all kinds of software projects, being standalone product, components and process improvements projects.

Bruhn NewTech has developed professional software for mission critical operations since 1987. Main products are various NBC (Nuclear, Biological and Chemical) warning and reporting systems marketed and sold internationally to military organisations within NATO.
Currently USA, England and Denmark are using our NBC warning and reporting system within all services nation wide.

This Software Process Improvement (SPI) experiment is part of the ESSI programme (ESSI PIE No. 27599). It was carried out in the periods from AUG 1998 to JAN 2000.

Starting scenario

Our software development has been structured around the classical "Water Fall" method. This development method lead to a number of shortcomings in product quality, however. Especially on the "soft" aspects of the software such as user friendliness and user support in terms of implementing the right functionality. And since user friendliness and user support are essential to Bruhn NewTech systems, which are often used in situations where the user is under heavy stress, Bruhn NewTech started experimenting using iterative aspects and user involvement in the development process.

The plans and the expected outcome

The plan for the experiment was to improve the software development process by adopting a DSDM (Dynamic Systems Development Method) [5] inspired RAD (Rapid Application Development) process, and customise it for our use.
Implementation of RAD, to the software process should ensure close co-operation with the users ensuring immediate feedback and thereby correct functionality of the released products, an essential element in the development of competitive products.

The expected improvements compared to previous reference projects were as follows:

- 30% reduction in change request in a period of 3 months after release.
- 30% reduction of requirements related errors encountered in the final system tests
- 20% reduction in the costs of producing project documentation
- All in all a substantially lower development effort and costs, without compromising productivity.

Due to the fact that RAD most likely would shift the focus towards user friendliness and usability, it must be evaluated whether or not the RAD method can coexist with the formal specification language VDM (Vienna Development Method) [2], to ensure the correctness and reliability of algorithms and mission critical components.

1.1 Limitations

A RAD process covers all disciplines in software engineering and management, meaning that developing and implementing a RAD based process involve changing all the employed software process elements:

- Project Management
- Requirements Management
- Design
- Implementation
- Test
- Etc.

We decided to focus our main experiment effort on the project management and the requirements management aspects of RAD projects although we had to adapt the other elements as well over the course of the baseline projects in order to maintain consistence with the RAD process.

1.2 Baseline projects

The experiments was carried out on three different baseline project, those are briefly described below.

Baseline Project 1 (BP1)

Content	: Adding minor customer specific requirements to an existing COTS Product
Team	: 1 Project Manager & 5 Software developers
Duration	: 5 Months

Baseline Project 2 (BP2)

Content	: Developing a component to be embedded within a larger system
Team	: 1 Project Manager & 2 Software developers
Duration	: 5 Months

Baseline Project 3 (BP3)

Content	: Adding a specialised module to an existing COTS Product
Team	: 1 Project Manager & 6 Software developers
Duration	: 4 Months

Implementation

1.3 Project organisation: Roles and Responsibilities

Although the main focus of the RAD process is the relationship between the development team and the user, the internal organisation of the project team needed to support the RAD principles as well.
To ensure the empowerment of the project team, and the support of the surrounding organisation, a new project management document was initiated: "Roles and Responsibilities", which defines the person/role responsible for all major tasks and project deliverables during the project period.

1.4 Timebox

Inspired by DSDM, the “Time box” concept was introduced as illustrated in *Fig. PGN.1*. The main purpose of this concept was to ensure that deadlines were kept, due to everybody keeping focus on what is most important in a given timebox. An essential element in the timebox concept is priorities, meaning that all tasks (e.g. implementation of requirements) in a given timebox must be prioritised. It must be possible to complete all tasks with the highest priority within a comfortable margin, but the timebox will most likely not be long enough to complete all tasks, this means that the most important tasks are completed and the lowest prioritised tasks are skipped. Because everybody has agreed upon this concept, and the right persons prioritise the tasks, everybody is satisfied with the outcome.

In our baseline projects, the length of the time boxes varies from 4 hours to 6 weeks depending of the task. The various requirements was usually time boxed with a duration between a few hours and up to a maximum of 2 weeks, however the project phases covering a prototype cycle was also handled as a timebox, and the duration for these was up to 6 weeks.

In the first baseline project the priorities of the requirements was not documented in writing but agreed upon between the developer and the customer directly whenever needed. In the second and third baseline project the requirements were all prioritised and this was documented as part of the requirement specification.

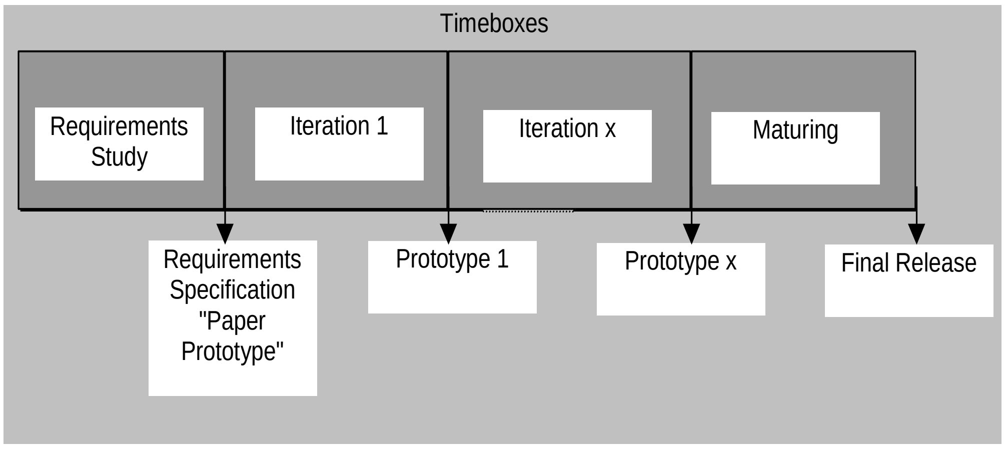


Fig. PGN.1: Iterative Development Process

1.5 Requirement management

The requirement management was enhanced in four different manners during the experiment.

Visual elements in the Software Requirement Specification

The Requirement management process was enhanced, due to the drawings of screen layouts in the SRS (Software Requirement Specification). These simple drawings (see: *Fig. PGN.2*) in the SRS have shown tremendous benefit when discussing new functionality with the customer, especially in cases where it was not possible to meet face to face.

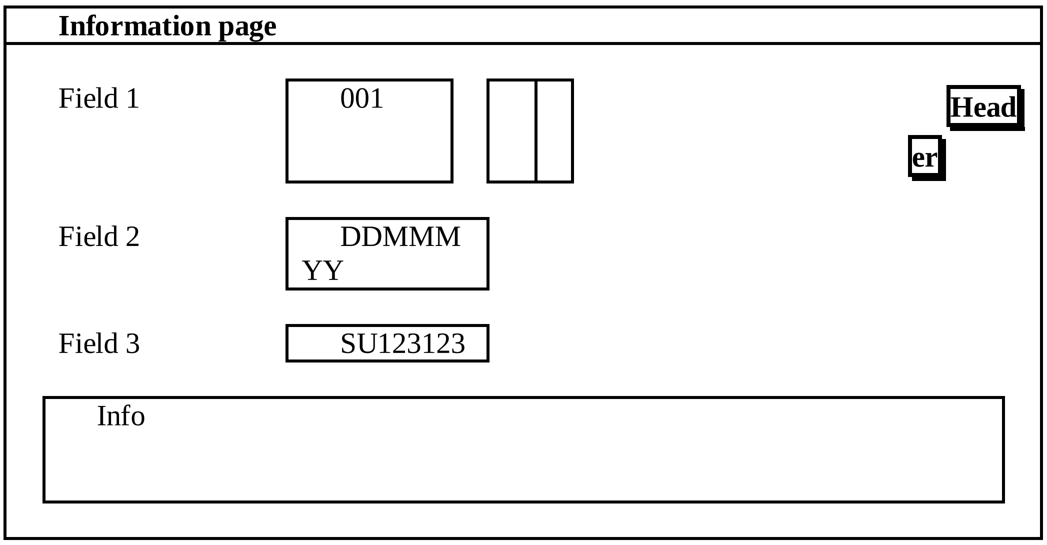


Fig. PGN.2 : Example of screen layout used in Software Requirement Specification.

Priorities

Another improvement of the Requirement management process was the implementation of documented prioritisation in the SRS. The following priorities were used:

Essential	:	System could not be accepted without having these requirements included
Highly desirable	:	System would most likely be accepted, but it will have a considerably reduced value to the user.
Desirable	:	"Nice to have" features which can be omitted if not enough time is available in the timebox.

Requirement study

An important part of the Requirement management is the "Requirement Study" illustrated as the first timebox in *Fig. PGN.1* above. This study is carried out as a workshop, with both the user and the developers, and the idea is to ensure that the developers having a solid understanding of the requirements at a very early stage of the project. The outcome of the workshop is documented in the SRS, including the first draft versions of screen layouts.

User involvement & Prototype meetings

The user involvement was done through a continuous dialog between the development team and the user representative. It was not possible to have the user integrated in the development team, however when new functionality was implemented to a certain "prototype" level, the program was demonstrated for the user, and the comments from these prototype meetings where documented and then evaluated and prioritised together with the remaining requirements.

These prototype meetings ensure that problems regarding unclear or inconsistent requirements are resolved up front rather than late in the project.

1.6 VDM

The VDM experiment was carried out on baseline project 2 (BP2). The aim was to introduce VDM, and evaluate the method for use in a small organisation like ours.

The work with VDM to supplement the requirements was not an immediate success. Although VDM is a strong tool, we observed little benefit in this particular experiment. The reason for this was primarily that BP2 turned out not to be well-suited for VDM-specification integration, but also, that we do not possess the critical mass of VDM expertise to implement and sustain VDM use within our organisation at this stage. Based on this we decided not to pursue VDM further in this experiment.

Result

1.7 Quantitative measurements

Objective: 30% reduction in change request in a period of 3 months after release, and 30% reduction of requirements related errors encountered in the final system tests

We have not been able to verify the effect of our improvements with respect to the objectives of reduction in the number of change requests and the number of requirements related errors. We have made an initial analysis of the reference data in order to set up the necessary criteria, but lack of effective metrics and post-experiment data prevents us from drawing any conclusions.

We did not have the necessary skills and experience required for developing the means of measuring and tracking the effect of the experiment on our main quantitative goals. We learned too late of more rigorous formal methods for error type categories (such as Beizer's categorisation scheme [1]), to apply this to our error registration and produce meaningful analysis of comparable data.

Objective: 20% reduction in the costs of producing project documentation

We can see now that the initial objective of reducing the documentation effort with 20% can not be achieved. We are developing more documents in order to support both the project management and the requirements management processes for our software development projects. We have developed new document types and significantly changed others in order to support the RAD principles employed in the resulting process. It is however the general feeling that the documentation produced now is necessary and adds value to the process, whereas it used to be regarded as a necessary evil with no particular purpose.

Objective: All in all a substantially lower development effort and costs is expected

Over the course of the development projects using the RAD principles implemented through RAMSES, we developed the feeling that we were more in control of the projects and of the development process, but when we examined the time registration data (see: *Fig. PGN.3* below), we were quite surprised by the results.

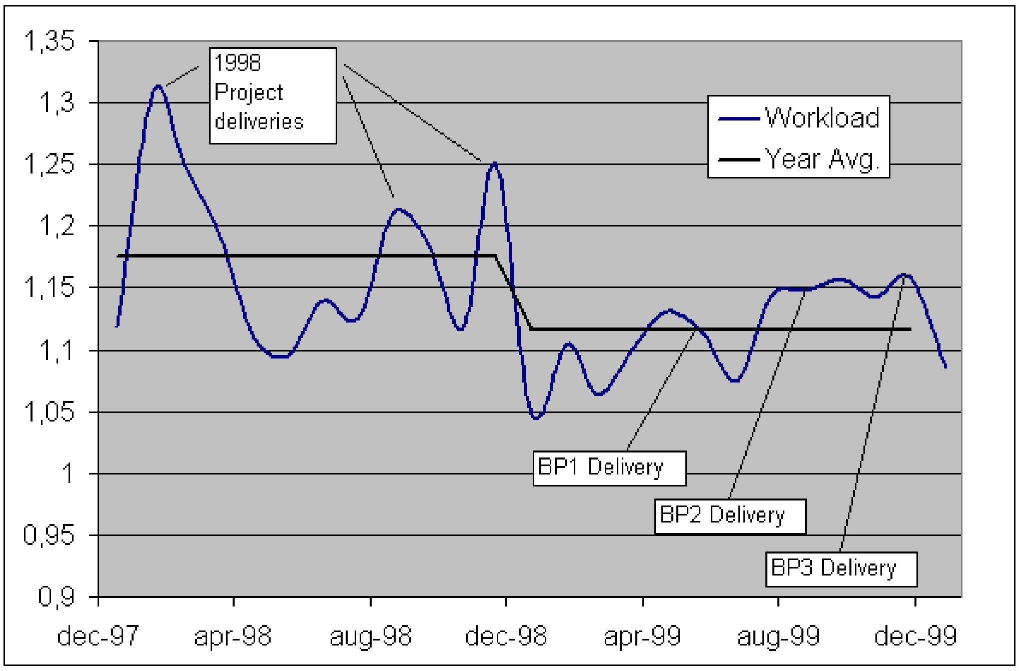


Fig. PGN.3 : Total development hours relative to normal working hours (37h/week) spent over the past two years (01-Jan-98 to 31-Jan-00).

We have reduced the workload peaks exceeding normal working hours at delivery milestones from 31% to 16 %, and the yearly average workload exceeding normal working hours from 18% to 12%.

As a very tangible benefit of the changes in overall workload, we have eliminated the need for overtime payments formerly used to manage the excessive peak workloads. We have previously needed overtime payment as compensation for peak working hours in conjunction with project deadlines. In 1998 we had an overtime payment comparable to the salary of 1½ software engineer, in 1999 we had no overtime payments at all.

We believe that the increased focus on user involvement, prioritised requirements and the timebox concept leads to better management of the customer expectations, and a better control over the entire project.

1.8 VDM

We conducted a workshop together with a consultant from IFAD (Institut For Anvendt Datateknik, Odense, DK) working on integrating VDM [2] and the VDM tools into the requirements process. We found that VDM is a demanding tool with great potential. At the time of the experiment we did not have sufficient in-house experience with VDM, and further use of the tool would require a significant training and implementation effort in the organisation, which was outside the scope of this experiment.

1.9 Qualitative measurements

Objective: Better management of customer expectations

The user representative participated in prioritising the requirements, which enabled a more qualified discussion with the customer regarding the consequences of making changes to requirements, and helps us to prevent requirements creep. Additionally the user involvement in the requirements elicitation and the use of prototype meetings has improved our ability to manage the customer expectations and improved their perception of the delivered product. We do not have a great deal of data to support this, but as part of one of the completed projects we sent a questionnaire to the customer to evaluate the conduct of the project and the customer satisfaction with the process and the end product. The customer was also asked to compare with former projects delivered under the old process. The results were very favourable to our new process. A further indication of the customer's expectations being met by the product is, that we have been asked to add new functions and make modifications, for which the customer is willing to pay. Whereas before, the customer may have been inclined to say that they had expected these functions to be part of the initial delivery and they should be included without additional cost.

Lessons learned

- One size does not fit all
  - We have to modify the DSDM/RAD principles to suit our purpose. It requires a learning phase and small-scale experiments to derive suitable processes.
  - The improvement effects arise from the tailoring of the processes rather than from adopting a general process model.
  - Overall process models such as DSDM provide good inspiration, but they do not go to the level of detail required, to achieve real benefits. It is probably less significant which framework (DSDM, MSF (Microsoft Solution Framework) [4] or RUP (Rational Unified Process) [3], etc.) is chosen, if the process work captures the necessary details for the real processes.
- The customer representative
  - Participation from the customer in a development project secures focus on the customer requirements, reduces "gold-plating", and gives a sense of ownership within the customer organisation, which increases the overall customer satisfaction.
  - The customer representative can become a bottleneck for resolving requirement issues if the requirements are not thoroughly analysed and described to a reasonable level of detail before development starts. It is important that the customer organisation is aware of the need for rapid feedback and intensive interaction regarding the requirements.
- Software process improvement pays of.

  - There is an immediate benefit from the analysis of the current status of the development process and the involvement of the development staff in process related activities.
  - Participation adds to the motivation and creates a company culture that supports continuous improvement.
  - By encouraging process improvement at all levels, it is possible to gain further operational efficiency from the knowledge and ideas of the employees regarding optimising the daily work routines, identifying bottlenecks and reducing time waste.
- Process improvement time must be scheduled explicitly in the resource plan
  - Customer project activities will always have a higher priority than process improvement, but this conflict can be avoided by explicitly scheduling the process improvement activities, e.g. using a fixed weekday for these tasks.
  - The external funding for process improvement secures top management acceptance and support for resource allocation.
  - Reporting and dissemination activities are demanding and time consuming. They require dedicated resources and scheduled time.
- Organisational impact
  - The process improvement experiment has served as a catalyst for starting an improvement culture. Through the work with the process improvement experiment it has become clear to everybody within the development department that it is possible to improve the processes, whereas earlier there was a lot of scepticism regarding the effectiveness of process improvement.
  - The entire development group has been involved in RAMSES activities. This has increased the awareness of the process improvement activities and facilitated the implementation of the processes in the baseline projects although the processes were not fully described.

Problems encountered during the experiment

Bruhn NewTech did not have a formal process improvement program in place before starting this ESSI PIE. That did cause us some severe problems in the early phases of the project, because we lacked the process and necessary methods to manage and track our software process improvement experiments. Key skills that we lacked were development of software measurements, tracking of results, and error analysis. Managing and conducting software process improvement activities requires a wide variety of skills and substantial in-house resources to support it.

As other projects, SPI projects are vulnerable to the loss of key personnel. We suffered the loss of two key persons early in the project, and thereby lost a lot of project continuity and SPI experience. One was the intended project manager, and the other an experienced consultant at our main subcontractor. As replacement for the intended project manager, we decided to break down the overall project responsibility and divide it between three persons. One with technical responsibility, one with administrative and EU responsibility, and one with dissemination responsibility. All three were at the same time responsible for other projects or areas and at times heavily overloaded. The result was lack of focus on the tracking of the experiment, so it was not a good solution to the resource problem.

With the lack of experienced SPI management within the organisation, we found it difficult to take the conceptual ideas of our original project proposal and converting them into a proper SPI experiment plan with the proper means of tracking, documenting and reporting of the results of the experiment.

We did not have the necessary skills and experience required for developing the means of measuring and tracking the effect of the experiment on our quantitative goals of reduction in requirements related errors and change requests. We should have drawn more on external experience from subcontractors, EU, or ESSI peers in resolving some of our problems early on in the project. We should have used early dissemination activities to share our problems and learn from others.

We failed to properly communicate the effort and the plans for the process improvement experiments throughout the organisation in the start of the project, which left parts of the organisation with the feeling that there was no progress.

Summary

We have during the RAMSES project (ESSI PIE No. 27599) gained useful experience with SPI (Software Process Improvement) and obtained results which documents that there are great benefits in SPI also for small companies.

The objective of the proposed experiment was to improve the software development process by adopting a formal RAD (Rapid Application Development) method.
We expected a reduction in requirements related errors and change requests of 30%, we expected a reduction in documentation costs of 20% and anticipated a substantially lower development effort and cost.

We have implemented a DSDM (Dynamic System Development Method) [5] inspired, RAD based process involving close interaction with the customer throughout the development process. We customised and developed the process and the necessary document templates to support the project team through a RAD project. We also improved our requirement analysis process and developed requirements specification documents supported by visual elements such as user interface designs in order to facilitate the discussions with the users before starting the programming.

The resulting overall process improvements have led to a more controlled development process and verifiably to the achievement of one of our original goals. The improvements have reduced the often very large workload peaks at delivery milestones by 50%, and thereby eliminated the need for overtime payments.

The involvement of the customers has resulted in realistic expectations to the final product and consequently more satisfied customers. We have not been able to verify the effect of our improvements with respect to the objectives of reduction in the number of requirements related errors and change requests. We have made an initial analysis of the reference data in order to set up the necessary criteria, but lack of effective metrics and post-experiment data prevents us from drawing any conclusions.

Our process improvement experiment provided some key lessons learned especially applicable to small software development organisations.

- Standard process frameworks (e.g. DSDM) must be tailored and detailed to the specific needs of the organisation.
- SPI management is a difficult discipline, and internal and external communication is very important.
- We did not have the necessary skills and experience required for developing the means of measuring and tracking the effect of the experiment on our main quantitative goals.
- Measurements must be planned as part of the experiment, and it requires specific skills and knowledge.

- We should have drawn more on external experience from subcontractors, EU, or ESSI peers in resolving some of our problems early on in the project, using dissemination activities to share our problems and learn from others.

In conclusion, we have achieved software process improvement with factual real value benefits internally and to our customers through conducting the process improvement experiments, and gained useful insights in the workings of software process improvement. We have encountered a number of problems during the experiment, mainly due to our lack of SPI experience. We have reduced both the average workload and especially the workload in conjunction with deadlines considerably without compromising the efficiency and the customer satisfaction. This is mainly archived by improved requirement management (User involvement and prioritised requirements), and by introducing the timebox concept.

The improvements archived during the RAMSES project are now included in our development process and used on all development projects when applicable.

References

[1] BEIZER, B. Software Testing Techniques. Second Edition.Van Nostrand Reinhold, New York 1990

[2] FITZGERALD, J and LARSEN, P.G., Designing Software: a model based approach, Cambridge University Press 1997

[3] KRUCHTEN, P., The Ration Unified Process, An introduction, Addison-Wesley Pub Co. 2000

[4] MSF (Microsoft Solution Framework), http://www.microsoft.com/technet/Analpln/process.asp

[5] STAPLETON, J., DSDM the method in practice, DSDM Consortium 1997

Recommended reading

This experiment is especially applicable to smaller companies, having problems in controlling the development process due to creeping/unclear requirements, and consequently delayed deliveries or increased workload in conjunction with releases or deadlines.

The experiment is completely independent of specific tools and platforms, due to the described enhancements are only related to the development process.

The following book is recommended for inspiration:

- STAPLETON, J., DSDM the method in practice, DSDM Consortium 1997 [5]

Bruhn NewTech - Company profile

Bruhn NewTech is an international company that excels in software and systems integration. We specialise in the development and maintenance of management information systems for operations in both the NBC battlefield and civil emergency response environments. NBC is an acronym for Nuclear, Biological and Chemical.

An NBC environment exists if Weapons of Mass Destruction (WMD) have been used or there has been a release, accidental or otherwise, of toxic material. Bruhn NewTech develops and supports NBC information systems that provide military commanders and emergency planners with rapid and accurate information to enhance vital decision-making.

Bruhn NewTech has 15 years of experience in the development and use of information technologies for Nuclear, Biological and Chemical defence purposes and emergency response. This has lead to a range of products, projects and training support activities for military forces, civil defence and emergency response planners.

For more information: http://www.newtech.dk/

Peter Gungaard Nielsen
Project Manager
Bruhn NewTech A/S

Education:

DEGREE	SCHOOL	DATES
B. Sc. Electronic Engineering	The Engineering College of Copenhagen	1991
Skilled Electronic Engineer from Nokia Data A/S Copenhagen.	-	1988

Professional Summary:
More than 8 years of experience in software development including requirement specification, design, implementation and test.
Responsibilities include resource management, project management of development projects and co-ordination and implementation of process improvements.

Work Experience:

1998 - Bruhn NewTech, Project Manager
Responsible for product development and process improvement projects.

1992 - 1998 Bruhn NewTech, System Engineer
Participated in software development projects. Tasks including all aspects of software development from requirement specification to final implementation and test.

1994 - Engineering College of Copenhagen, External examiner
External examiner within the area of OOA, OOD and C++

1992 - 1992 Engineering College of Ballerup, Teacher
Teaching in design and implementation of minor SW application and analysis of analog electronic circuits.

1991 - 1992 Engineering College of Copenhagen, Teacher
Teaching in design and analysis of discrete and integrated electronic devices

Svend Skafte
Systems Engineering Manager
Bruhn NewTech A/S

Education:

DEGREE	SCHOOL	DATES
B. Com. Bachelor of Commerce	Copenhagen Business College	1995
M. Sc. Software Engineering	Technical University of Denmark	1991

Professional Summary:

Mr. Svend Skafte has 10+ years of experience in software development, software integration, communication interfaces, software quality management and project management.

Responsibilities include requirements analysis, system design, programming, and program management (proposals, planning, organization, control and customer interface).

Work Experience:

1994 -	Bruhn NewTech, Systems Engineering Participated in major software development projects. Project manager for product development, and program manager for systems integration projects. Since July 1998 Systems Engineering Manager, head of the systems engineering department in Bruhn NewTech A/S. Overall responsibility for the software development organization, personnel, development processes and software quality.
1991 - 1994	A/S MODULEX. Software Engineer. Participating in the development of turnkey passenger information systems for airports, railways, bus terminals, etc. Project manager for a large passenger information system for British Rail.
1990 - 1991	Institute of Systems Science, National University of Singapore. Visiting Scholar. Teaching and research in knowledge engineering and artificial intelligence.
1989 - 1990	Soudronic AG (Switzerland). Knowledge Engineer. Development of an expert system for real-time inspection of glass bottles.
1987 - 1989	Electromagnetics Institute, Technical University of Denmark (DTU). Software Engineer. R&D in Expert Systems and Artificial Intelligence. Development of knowledge based system for assessment of thermal indoor climate in office buildings.

Software Quality Management and Software Process Improvement in Denmark

Karlheinz Kautz,
Copenhagen Business School, Department of Informatics
Howitzvej 60, DK-2000 Frederiksberg, Denmark
Karl.Kautz@cbs.dk,
Tel.: +45 38 15 24 00
Fax.: +45 38 15 24 01

Faisal Ramzan
Midtermolen 1, P.O. Box 2662, 2100 København Ø, Denmark
faisal.ramzan@dk.arthurandersen.com
Tel: +45 35 25 25 25
Fax: +45 35 25 20 01

Abstract

The software business is a fast growing industry sector and lack of quality has significant consequences for society and economy. However reports about unfinished development projects, project overruns and system and software failures are still the rule. Software quality management (SQM) standards and software process improvement (SPI) methodologies are all promoted to solve these problems. However little is known about how wide spread these standards and methodologies actually are. We have therefore performed a questionnaire based survey in the Danish software producing industry. The standards and methodologies are not known by 40% of the responding organizations and only 36% of the enterprises use to a different degree one or several of the approaches, while two thirds of the organizations that had not adopted any of the approaches had never heard about them. These and further results concerning the effects of utilizing the standards and problems with their introduction will be presented and discussed in detail in the paper.

1. Introduction

Software development is a complex process, which covers various activities until the final product is completed. Many organizations have problems and project overruns and defective systems are still the rule.

On might share the opinion of Denmark's minister for work and employment, who recently commented that the disaster[1] with Denmark's job center system Amanda was no reason for unease as missing functionality and exceeding budgets are an inherent element of software production ([1]).

However, one might be nevertheless concerned as software plays a more and more important role in private and professional life. According to Zahran ([2]) in the last years around 4000 people lost their lives due to software errors and in the recently emerging area of e-commerce, a mistake in an electronic payment system can have fatal consequences for organizations, which invest in such technology.

Instead of ignoring these problems, it might therefore be important for software producing companies to know where their difficulties lie and how they can solve them.

After numerous attempts to improve the situation with technical approaches ranging from structured programming over CASE tools to object orientation, managerial approaches based on the insight that a product can only be as good as its development process and thus directed towards both product and in particular process have gained growing attention. They are generally known as software quality management (SQM) standards and/or software process improvement (SPI) approaches [3].

In Denmark, single studies concerning their utilization by selected organizations have been performed ([4], [5]). No further spreading of these measure will however happen, if we do not know how wide spread knowledge and actual application of these approaches are, what the concrete reasons for their adoption and non-adoption are and what their perceived effects are. But no such study exists in Denmark and this is the objective of the investigation reported in this paper.

The paper is organized as follows. In the next section the research framework is explained and then the results organized in sections concerning demographic data, awareness and utilization of SQM and SPI, SQM and SPI and the perception of software development problems, introduction of SQM and SPI, and objectives and effects of SQM and SPI. Finally, the main conclusions of the study are compared with similar work and further research questions are outlined.

2. Research Framework

Contents of the Study

The study, which was performed in 1999, deals with knowledge and utilization of SQM standards and SPI methodologies. It covers the ISO9001 standard [6] for quality management systems and its guidelines for the implementation in software organizations as well as the especially developed TickIT scheme [7,] which gives even more precise details about the realization of these standards based on an initiative of the British Computer Society and British Standards Institution.

Furthermore the staged capability maturity model (CMM) [8] developed at the Software Engineering Institute (SEI) in the USA and its European counterpart BOOTSTRAP [9], which is based on the ISO-standards and the CMM are covered by the study. We were also interested in the diffusion of SPICE [10], a standard for software process assessment and improvement methodologies. In addition, the respondents could put down other wide-ranging, also self-developed, methodologies, which they used for comprehensive quality management and software process improvement.

[1]According to Politiken, one of Denmark's largest daily newspapers, 18.05.2000, the project has a schedule overrun of over 60% and a budget overrun of more than 70% (ca. 23 mill. $).

Finally, we also asked about the use of individual improvement measure such as evaluation and amendment of planning, tracking, control and review activities and single quality assurance techniques like structured walk-throughs of specification documents, unit, integration and validation testing as opposed to or as part of the above named all-inclusive standards and process improvement methodologies.

The study does not take in metrics-driven approaches (see f. ex. [11], [12]). They are not based on formal audits and/or assessments of the current practice and problems and do not include an explicit model for improvement. As such they are different from standards and model-based approaches. Therefore, although they might be valuable supplements for software quality management and software process improvement, they are not considered here.

Research Method

To answer the research questions a survey instrument with 64 questions was developed and tested in 3 cycles with representatives of the target population. Both nominal and ordinal scale variables were applied and where necessary, groupings of answer categories were made. The data was statistically evaluated and to secure the significance of the results 3 tests – a chi-square, a likelihood ratio chi-square, and a MH chi-square test – were performed and a significance level of 0,05 was chosen [13].

The questionnaire was sent to 580 organization, either to the managing director, the IT manager or the quality manager, where known. The 80 member companies of the Danish Computer Technology Forum all received a questionnaire. The remaining 500 companies were chosen from 2 databases for organizations, where software developers are employed or which provide software consultation. From a composed list of 1100 organizations, which contained a limited number of enterprises with under 5 and under 10 employees – these had been the largest portion of the originally over 8000 registered firms – 500 were randomly chosen. As 33 organizations declined to take part in the study and out of the 118 responses 7 could not be used, with 111 answers usable we achieved a response rate of 20,3%, which is quite satisfying.

The responses fall into 2 categories, namely responses from the IT sector, companies, which consider software as their primary product, and others, which develop software as part of their primary product or service. In the representation of the results, we will relate to this distinction, wherever it is significant.

The population can also be divided into two other categories, namely those, who use the standards and/or improvement approaches and those, who do not utilize them. To investigate further the reasons for adoption and non-adoption the survey respondents were asked to indicate whether they were willing to participate in an interview study. As a result 6 single case interviews with 6 representatives of 6 companies, 3 using quality management and/or process improvement, and 3 not applying these approaches, were conducted on the basis of an open, theme-oriented interview guide. Each interview lasted 90 minutes and its contents was verified by the interview objects. The results are not used for generalization, but to support some of the correlations, which the survey only could touch upon on the surface. The method triangulation supported the fulfillment of the requirements concerning representativity, validity, and reliability, which have to be posed for a descriptive and explanatory investigation like ours [14].

3. Demographic Data

Industry Sector

As stated 111 organizations participated in the investigation: 72% (81) consider themselves as part of the IT sector, 13% are from industry, 6% from banking and finance, 4% from telecommunication and 5% from unspecified other sectors.

Size and Product Portfolio

The majority (67%) has more than 50 employees, however in the IT sector 42 organizations have under 50 staff, whereas this is only true for 2 of the other organizations. As the response rate of organizations with under 25 employees is quite low, some of the conclusions, where size plays a role, have to be considered critically.

In the IT sector, otherwise, 80% of the companies have a software development department, whereas the number is 97% for the others. With regard to the size of software development projects and project groups, 66% of all companies report that their projects are under 3 man-years, in the IT sector the number is 73%, for the rest it is 47%. The project groups have a size of 1-5 in 55% (61) and 6-20 employees in 38% (43) of the cases. Only 6%of the organizations have projects with more than 21 project members. In the IT sector 61% (50) have project groups with 1-5 members, in the other sectors this number is 37% (11).

Of the non-IT organizations 33% (13) deal with the realization of standard products for the market, 28% (11) with tailor-made customer products and the same amount with development for company internal purposes. The IT sector is dominated by adjustment of standard products (38%) and the development of tailor-made customer systems (35%).

The data shows that larger organization regardless of their sector have larger development departments. Larger enterprises have larger projects in terms of staff and man-years. A possible explanation for the differences comes from the interviews. They show that non-IT organizations have projects of larger size and more man-years. When a product has been finished, it has to be maintained and further developed. This seems to happen in the same project rather than in a new development effort.

The results are also evidence for that larger organizations develop more frequently for internal use in the organization.

Respondents' Profiles

The respondents are experienced staff: 83% are older than 36, 56% have been employed longer than 6 years in the company, 69% have been in their actual position longer than 3 years, 83% have over 6 years experience with software development and 51% have performed more than 20 development projects. With reference to quality management 57% have 3 or more years of experience; 35% (39) are managing directors, in the IT sector this number is 46% (37), and for the7% (2).

Also some other results concerning the differences of respondents from the IT sector and the rest are interesting. Relating to performed development projects 60% (48) in the IT sector have performed more than 20 development projects, for the rest the number is only 28% (8). This might be linked to the size of the projects, which is smaller and shorter in the IT sector. This assumption is confirmed by the numbers concerning development experience expressed in years: 77% (23) respondents from non-IT organizations have over 11 years of development experience, in contrast to 47% (38) in the IT sector. Concerning age and experience with quality management 44% (35) of the respondents from IT companies are over 41 years old, for the rest the number is 67% (20) and more than 50% (15) of the respondents in this segment have more than 6 years of experience with quality management. For the IT sector these are 26% (21). A possible explanation for this might be that the non-IT organizations deal and have dealt with quality management longer (and) in other than their software departments.

4. Knowing and Using Software Quality Management and Software Process Improvement

Knowledge about SQM/SPI

Knowing about SQM and SPI is a prerequisite to utilize them. However 40% (44) organizations do not know either the ISO9001 standard, the TickIT scheme, the CMM, Bootstrap or SPICE. For the IT-sector this number is 51% (41), for the rest 10% (3). All others know at least one methodology.

Concerning the single methodologies the CMM is most known: 52% know it, followed by Bootstrap with 40% , the TickIT/ISO scheme 34% and SPICE with 26%. The study also shows - although not statistically supported - that organizations, which use one methodology know about others. Out of the 40 organizations, which use SQM/SPI 30 know the CMM, 23 the TickIT scheme, 22 Bootstrap, and 17 SPICE. For the 70 organizations without SQM/SPI the numbers are lower: 28 know the CMM, 22 Bootstrap, 15 the TickIT scheme, and 12 SPICE. This indicates that the organizations base their decision to introduce a particular SQM/SPI methodology on a conscious evaluation process and knowledge about alternative approaches.

There is, however, a clear need for the Danish IT producing organizations for information: 71% give as a reason for their ignorance missing information, whereas only 11% have no interest and 9% see no relevance in SQM/SPI for themselves.

Utilization of SQM/SPI

There exists an overwhelmingly positive attitude towards SQM/SPI methods[2], which is expressed by 87% of all answering general managers, 86% of the IT managers, and 73% of all answering staff. This is probably due to the positive connotation that the concept quality has.

However 64% (71) of the responding organizations are not using SQM/SPI. As reasons they provide that SQM/SPI is too resource demanding (25%), that staff is lacking training and education in the area (24) and that information is missing (21%). Whereas 30% have not at all considered SQM/SPI and 8% think it is irrelevant for them, 8% are planning and 54% are considering it. These do so because, they want to identify the weaknesses of their processes (32%), experience to many mistakes (25%) or have a request from management (20%).

Concerning single improvement techniques, one or more of them are used by 89% organizations and stand-alone quality assurance in form of testing are applied by 93% of all organizations, structured walk-throughs by 78% and monitoring of functionality (acceptance test) by 71%. There are no significant differences concerning the use of these techniques with regard to the different sectors or the other characteristics of organizations with and without SQM/SPI.

The study shows also that 36% (40) of the responding organizations use SQM and/or SPI: 24% are ISO certified, 22% use Bootstrap, 18% CMM, 14% use a methodology they have developed themselves, 14% name methodologies, which we had not asked for[3], 6% are TickIT certified and 2% use the TickIT scheme, but without certification, 9 organizations use more than 1 approach.

On a first glance these numbers give the impression that SQM/SPI is already widely used in Denmark, however looking at the IT sector and the rest separately reveals that

in the IT-sector only 28% (23) use SQM/SPI, whereas 72% (58) do not; 23 use methodologies, which fall into the category ‘other’ and 8% use self-developed approaches, 19% are ISO certified, 19% use CMM and 15% Bootstrap; 4% use more than one approach

from the rest 32% (8) are ISO certified, 28% (7) use Bootstrap, 20% (5) have an own developed methodology, 18% (4) use the CMM and 4% (1) something else; 20% (6) use more than one approach.

Given the fact that deploying the standards and methodologies completely is an extensive and resource demanding endeavor two explanations are possible: SQM/SPI is really wide spread in Denmark or not all organizations apply the approaches in the formally pre-and described way. The interview results support the second explanation: Organizations adjust SQM and SPI standards and methodologies according to their own needs instead of applying them completely as described in the literature. Thus, the number for those applying the methodologies fully is almost certainly lower than 40.

Characteristics of Organizations with SQM/SPI

[2] The numbers are actually even higher for the utilisation of software development methodologies and tools, but this is not further discussed here.

[3] We did not investigate the features of these methodologies any further.

Larger organizations and larger software development projects measured in man-years have a larger tendency to utilize SQM/SPI, whereas the size of the development department or the project groups does not play a any role.

An explanation for this might be that larger organizations have more resources and therefore have better possibilities to engage in SQM/SPI. This does however not mean that SQM/SPI is only be used by these enterprises. As a matter of fact 61% (14) of the organizations in the IT sector with under 100 employees use SQM/SPI and 30% (7) of the organizations in this sector with SQM/SPI have under 50 members of staff. Thus SQM/SPI appears to be suitable also for smaller organizations.

The IT sector has less experience with SQM and/or SPI compared to the rest: over 50% (14) organizations in the IT sector have under 4 years of experience, the number for the others is 25% (4). While over 50% (8) of the non-IT organizations have more than 8 years of experience in the area, in the IT sector these are only 22% (5) firms. This is probably due to the fact that many non IT organizations have a long history of quality management with regard to the production of their primary product.

An own quality department have 77% of the non-IT and 57% of the IT organizations and they are larger in the non-IT sector. This might be explained by that for these the quality department also works with quality management for their primary product or service.

Concerning a defined development process 90% of the organizations with SQM/SPI answer that they have a defined process, for the organization without SQM/SPI this number is 39% enterprises.

5. Software Quality Management, Software Process Improvement and the Perception of Problems

Problems with Software Development

Software quality management and software process improvement are intended to solve the problems within software development. We were therefore interested in the respondents' general perception of problems in software development and the conception of the relationship between problems and SQM/SPI. In particular we looked at the following technical and support activities: requirements analysis and specification, system design, programming, testing, documentation, project management, utilization of development standards, configuration management, quality assurance and change management (adjustment of project plans due to changed customer requirements). The most significant results are:

- requirements specification and change management are the two areas, which clearly stand out from the others: 25% (28), respectively 24% (26) of all respondents conceive these areas as very problematic
- requirement specification, testing, documentation, project management, quality assurance and change management are all areas, which are regarded problematic or even very problematic by over 50% of the respondents
- programming is considered by 68% (75) as creating little or no problems
- development standards, system design and configuration management are seen by 57% (63), 52% (57) and 51% (56) as activities with little or no problems.

In general, the non-IT organizations perceive the majority of the activities as more problematic as the IT sector.

There is also a clear dependence between the perception of problems in the various areas. Those, who experience requirements specification and systems as problematic actually see all other areas as problematic. For all other activities there are also dependencies between them and all, but 1 or 2 areas.

Size and Problem Perception

In the context of quality management and process improvement size is often used as a variable explaining adoption and non-adoption. Concerning the perception of problems it can be said that larger organizations have a tendency to experience, requirements specification, testing, documentation and configuration management as more problematic than small organizations, in the IT sector large organizations see especially requirements specification as more problematic than small organizations, and there is a tendency of large project groups viewing change management as more problematic than small project groups. This seems understandable as large groups probably have to use more effort to co-ordinate their work.

In summary, there is however only little correlation between the organizations' size and their perception of problems and no correlation between the size of software development departments, of project groups and of development and the respondents perception of the organizations' problems with software development.

Using SQM/SPI and Problem Perception

SQM standards and/or SPI approaches are used by 36% (40) organizations. These have the apprehension that the activities within software development are more problematic than those organizations have which do not use these measures. This is in particular true for requirements specification, system design, testing, documentation, project management and configuration management. There is also a tendency for development standards and change management, but the connection is not statistically supported. The only area, which is considered as creating little or no problems by more organizations with than without SQM/SPI is quality assurance.

For the IT sector separately, the correlation is even stronger and shows that for this sector the difference between those enterprises, which use quality management and those, which do not is even bigger.

We also asked the organizations whether they used stand-alone quality assurance measures and single improvement techniques: over 70% do so, but their utilization has no significance for the organizations' perception of problems within software development.

SQM/SPI as Creators of Awareness

There are at least three possibilities for the differences concerning the perception of problems:

- Organizations that have introduced quality management have had larger problems with software development and therefore started e deploying quality management and process improvement.
- Organizations that do not use SQM/SPI know that they do not have problems and therefore they are not interested in the standards and approaches.
- Organizations that do not utilize quality and process measures do not realize that they have problems with software development, whereas organizations with SQM/SPI are more conscious about their problems within software development.

The last explanation seems to be the most probable: 46 of the organizations without SQM/SPI have plans or are considering these measures; 37 of them do so, because they want to find the weaknesses in their development processes and see SQM/SPI as adequate methodologies for this purpose.

Thus, the organizations see SQM/SPI as an approach that helps them find their problems and as such gives them an overview over their software development activities.

Our investigation shows that both organizations with and without SQM/SPI have problems with software development. While some of them do not try to solve these problems, because they are not aware of them, others are very conscious about how to work with their difficulties.

The 3. explanation is also supported by the fact that SQM/SPI according to the majority of the respondents has an overall positive or very positive effect on all the different activities in software development and this although organizations with SQM/SPI perceive software development as more problematic than those without.

Thus, SQM/SPI can be used to get an overview over the problems in software development and this contributes to a positive change of the single activities and the processes as a whole.

It can be argued that the positive effects actually confirm the first rationalization, but this is only a part of the explanation. The interview results support also that organizations with SQM/SPI are more conscious about their situation, because SQM/SPI provides the opportunity via evaluations and assessments to focus on the actual status.

In comparison, organizations without SQM/SPI are used to what they do without assessing and reflecting the situation. The existence of problems with requirements specification or project management for them is a natural element of software development.

It is not necessarily a formal assessment or evaluation, which is needed to become aware of the problems with software development, but the insight that its possible for the organization to improve is important. Such an insight is however often blocked by an understanding of quality management and process improvement as demanding extensive documentation, being bureaucratic and impeding creativity. Another reason might be that the business world just accepts that the majority of software projects overruns budgets and time schedules. Finally, as long as staff are evaluated according to whether they deliver on time as opposed to that they deliver quality, SQM and SPI will not be recognized as relevant. These attitudes have to be changed before SQM/SPI can be seen as ways to improve process and product quality.

6. Introducing Software Quality Management and Software Process Improvement

Initiating and Implementing SQM/SPI

The main reasons for starting to work with SQM/SPI are the organizations' fundamental interest in quality (25%, 22), the perception of too many mistakes (22%, 19) and a demand from top management (18%, 16). The initiative is in over 70% of the cases taken by management - top management, software management, project management. Whereas in the IT sector top management, may be due to the short distance between management and development scored highest (30%), for the rest the software managers (44%) are the main initiators.

The study shows that 53% of the organizations use external assistance for the introduction of SQM/SPI. This help is primarily used for assessments and evaluations, launch and establishment of a quality or improvement project and for consultation.

As a starting point 50% of all organizations choose to implement SQM/SPI in selected projects, while 13% start with selected departments, and 29% decide to start with a company wide implementation.

There is however no correlation between the activities, which are perceived as being problematic and those, which the organizations start their SQM/SPI activities with.

Project management, requirements specification, testing, development standards and system design are those areas, where improvements start. These are subsequently also the areas, where the introduction of SQM/SPI has gone longest and is considered as implemented.

The chosen implementation strategy - not to start company wide - indicates that most organizations are aware of the complexity of the introduction process. This is confirmed by the respondents' answers, where 70% (28) grade the process as difficult or even very difficult.

Problems of Implementing SQM/SPI

During the introduction process missing resources (36%), lacking staff engagement (24%), lacking engagement of project managers and missing management support (12%) are seen as the main sources of problems and as a solution 16% think of management support, while 34% mention staff and 25% project manager training and education. As a consequence 69% send their staff to training with regard to SQM/SPI[4]. In the IT sector the number is 78% , whereas for the rest it is 56% . However, these organizations use more days for training than the IT sector. The difference between the IT sector and the rest can be explained by the fact that only 41% non-IT organizations see training as a solution as

[4] The number for training in methodologies to support the technical activities are however higher.

opposed to 87% (20) in the IT sector. May be the staff in non-IT organizations knows more about SQM/SPI on beforehand due to the organizations’ longer deployment of SQM in non-IT departments.

SQM/SPI and Organizational Change

The implementation of SQM/SPI also leads to changes in the organizations in 63% (25) of the cases; the number is a little bit higher for the IT sector, where 65% (15) of the organizations made changes as compared to 59% (10) of the rest. The introduction of SQM/SPI thus means organizational change.

The survey does not investigate the kind of changes, but the interviews show that the changes are of different character: some are changes of the physical and/or organizational structure, others are changes in the business and development processes.

They all have far reaching consequences on the organizations as whole and are not considered as isolated solutions to local problems. As such the introduction, implementation and utilization of SQM/SPI means organizational development.

7. Objectives and Impact of Software Quality Management and Software Process Improvement

Effects of SQM/SPI

The study shows that the effect on the organizations’ business variables based on a subjective judgement is perceived as positive or even very positive. This is valid for customer satisfaction (79%), staff satisfaction (73%), delivery precision (62%), staff motivation (52%), resource consumption (50%), cost reductions/expenses (43%), and budgeting precision (36%). Actually only in 6 cases the effect is judged to be negative.

Concerning the technical and support activities for software development the numbers are similar.

Positive or very positive effects are stated in the case of requirements specification (81%), system design (53%), programming (62%), testing (86%), documentation (69%), project management (75%), configuration management (61%), use of development standards (55%), quality assurance (70%), and change management (47%), only in 4 cases the effect was negative.

Another effect is that a fourth of the organizations (24%, 17) plan to work on more new processes. This shows that they are not only focusing on resources, but more on the new processes SQM/SPI bring about. Still, 20% (14) of the organizations intend to use more resources for SQM/SPI, and 27% (19) will work upon that SQM/SPI thinking pervades the whole organization.[5]

Defining Aims and Measuring Achievements

All the above numbers are as mentioned based on subjective assessment and in the case of expenses, resource consumption, and budget precision nearly a third of the respondent can not make any statement about the effect at all. The same, although a little bit less with 1/5, is true for the activities system design, utilization of development standards, quality assurance, and change management.

The study shows that all organizations with the exception of 1 have formulated business related objectives in at least one of the following areas: customer satisfaction (20%), staff satisfaction (20%), delivery precision (19%), staff motivation (8%), resource consumption (9%), cost reductions/expenses (13%), and budgeting precision (9%).

In total, 90% of those using SQM state that they perform measurements. These are distributed as follows: 20% of all answers fall into the category of delivery precision, 18% measure customer

[5] The respective numbers for organizations without SQM/SPI are 35% (44) and 19% (24).

satisfaction, 17% staff satisfaction, 14% expenses, 11% resource consumption, 10% budget precision and 9% staff motivation.

With regard to objectives for the software development activities 12 out of 40 organizations have not formulated any objectives and only 23 organizations perform measurements. The actual distribution is: testing (objectives: 14%, measurements: 14%) documentation (9%, 13%), system design (7%, 7%), project management (9%, 13%), programming (8%, 7%), quality assurance (12%, 12%), requirements specification (12%, 12%), configuration management (7%, 8%), development standards 10%, 8%), change management (4%, 5%).

The study has not investigated, which objectives the organizations' have defined and how they perform the measurements. But, it has not been possible to find any correlation between the organizations' defined objectives and their actual areas of measurement.

All this indicates that measurement in general is a very problematic area for all organizations. This suggestion is confirmed by the interview results: none of the organizations has a somehow functioning metrics program. However, this does not mean that organizations with SQM/SPI do not engage in measuring. They see it as a means to improve their processes, but have problems with defining the appropriate metrics, whereas organizations without SQM/SPI do not see the value of measurements.

8. Summary

The results of the study have to a certain extent been discussed in the above sections. Here we like to summarize them, put them into a context of related work and point out themes for future work.

All most all organizations have a positive attitude to SQM, but SQM standards and/or SPI methodologies are NOT known by 40% (44) organizations.

The positive attitude towards quality management is probably due to the positive connotation, which the concept of quality has. However, knowledge about SQM/SPI is not especially wide spread in Denmark. This is inline with the results from a similar survey in Norway [15], whereas in a survey from 1995 [16] comparing Italy, the UK, Germany and France more than 75% of the respondents are not aware of the SPI approaches, but only about 20% do not know the ISO9000 standard series .

SQM and/or SPI are used by 36% (40) organizations.

The study reports that 36% (40) organizations utilize SQM and/or SPI. We have already discussed that the actual number of organization might be lower as most organizations have adjusted the standards and methodologies to their needs or have developed their own approaches. In this light the 56% utilization reported from Sweden[17] look quite high. We do, however, not know the definition of the term SQM underlying that survey. In Norway [15] 17% use the TickIT/ISO scheme, whereas 8% use Bootstrap, the CMM or own developed methodologies. In Sweden the TickIT/ISO scheme and the CMM are most spread, while SPICE is more used than BOOTSTRAP. In Denmark ISO 9001, Bootstrap and the CMM are most used, and SPICE is not used at all. This difference can probably be explained by the fact that a particular Swedish change agency pushes the ISO standard and SPICE, whereas in Denmark one agency is very active promoting Bootstrap. The accumulated numbers for Italy, the UK, Germany and France are: 33% use the ISO9000 standard series for software activities, 28% the CMM – due to an extensive utilization in France (69%), 8% SPICE – again due to a very large application in France (28%), and 8% Bootstrap

SQM and /or SPI increase awareness for software development and its problems.

There exists a wide spread understanding that SQM and/or SPI create increased attention concerning the software development processes (see f. ex. [9], [10], [18]). Methodologies like the CMM explicitly aim at providing a more sophisticated picture of the development activities in accordance with the ascending level of maturity of the organization. Our study confirms that the awareness of software development grows with the use of SQM and/or SPI. In particular the study concludes that organizations, which utilize SQM and/or SPI are more aware of their problems - and as such they have an appropriate basis for improvement - than organization without SQM/SPI. In a more recent European study [19] about problems in European software production units, this supposition is also put forward, but no evidence is provided.

SQM and /or SPI have a positive effect.

In contrast to [19] where the conclusion is that SQM/SPI do not lead to a reduction of problems in software development, the respondents in this survey state that SQM/SPI have a very positive effect on both business and software development factors. This is in line with large parts of the literature (see f. ex. [2], [5], [8], [9], [20]). It has however to be considered that the survey has predominately been answered by management, who in many cases are the initiators of the SQM and/or SPI endeavors and that no assessment results and/or hard measurements have been required as evidence for the respondents' subjective judgement.

SQM and/or SPI is to a greater extent used by larger organizations, but can be used successfully by organizations of all size.

The study shows that larger organizations have a greater tendency to utilize SQM and/or SPI than smaller organizations. At the same time the investigation indicates that both large and small organizations adjust the standards and the methodologies to their own needs. This is confirmed by f. ex. [5] and [21]. The study of the diffusion of SQM /SPI in Norway [15] provides results, which also support this research. There the conclusion is that larger organizations apply SQM/SPI to a greater extent than smaller ones. It is actually argued that it is uncertain whether small companies will adopt methodologies like the CMM and Bootstrap as these seem to better fit the needs of larger firm. However, although these methodologies originally were aimed at large organizations, [22], [23], [24] and [25] all discuss SQM and/or SPI in relation to small enterprises and show possibilities how these can be beneficially used by smaller firms. The suitability of SQM and SPI for organizations of all sizes is corroborated by the fact that regardless of size most organizations report a positive effect of the measures.

Lack of resources and missing staff engagement are the biggest problems when introducing SQM and/or SPI.

Lack of resources is in general a problem when introducing novelties, especially in small and medium-sized organizations [26]. The Danish software producing units are no exception here and confirm the results from the Norwegian study [15]. Missing staff engagement might appear as a surprise, but as mentioned above the majority of the respondents are members of management. The survey does not ask for the reasons of the personnel's little engagement, but it be a result of insufficient empowerment and encouragement for employee participation. Active involvement, however, has long been recognized as one of the decisive factors for successful implementation of SQM and/or SPI programs [27].

Measurements and metric programs as part of SQM and/or SPI are a major problem.

Measurement is a complex process and the organizations in the study have major problems with metrics programs as a part of SQM and/or SPI.. This confirms the literature in the field (f. ex. [28], [29], [30], [31]).This literature contains some advice, but this seems to be seldom used. The Swedish investigation [17] also substantiates this conclusion. It shows that measurements and metrics are considered as most problematic in comparison to other factors, which constitute SQM and SPI. The Swedish study also includes a European wide comparison and also there measurements are seen as the most problematic area of SQM and SPI.)

SQM and/or SPI means organizational development.

This conclusion of our investigation is underpinned by the literature, which confirms that SQM and/or SPI are organizational development (see f. ex [2], [5], [8], [32]) that has consequences beyond the software producing units of the enterprise for the organization as a whole. In our study more than half of the answers state that they performed, respectively, had to perform structural changes in their organization when they introduced SQM and/or SPI. While this insight might be well known and documented in other fields of management and in systems development (see f. ex, [33], [34]), in the context of SQM and/or SPI in practice, companies do not always seem to be prepared as the problems with the introduction of the standards and methodologies, which are reported in our study, show.

9. Future Work

This investigation was the first of its kind in Denmark and one of a few in the area of diffusion and adoption of SQM/SPI internationally. As such it creates a basis for academics, practitioners, and organizations that are engaged or want to engage in work with SQM and/or SPI. Although the results are grounded on a solid statistical basis and method triangulation, some problems like f. ex. the relation between the respondents' positive attitude towards quality and the, in this context, comparable low deployment of quality management standards and improvement approaches and the serious problems concerning measurements and metrics have not been anticipated during the design of the research and have thus not been studied thoroughly. Here further in depth research is needed. The results of the study also raise a number of other unanswered questions. We will briefly present four of them, which should be subject of further research:

- Small organizations do not seem to utilize SQM/SPI as regularly as large organizations. It would be interesting to investigate further why this is so.
- Organizations with SQM/SPI perceive software development as more problematic than those, which are not using these approaches. Here it is obvious to take a closer look at the individual activities and to analyze the differences in handling the various tasks.
- Most organizations believe that training and education can solve the problems during the introduction of SQM/SPI. It would be interesting to investigate, which particular qualifications are needed by management, project leader and staff for SQM/SPI.
- Finally, the study reinforces that SQM/SPI is organizational development. It is therefore obvious to investigate in more detail, which influence SQM/SPI have on an organization, in particular its business processes.

The study indicates a need for an increased research effort in the areas of software quality management and software process improvement. However, there is first and foremost a need for the spreading of information – and education - about SQM and/or SPI. Most of the organization without SQM and/or SPI do not know which standards and methodologies exist. Without such knowledge it is difficult for them to start software quality management and software process improvement work.

Acknowledgements

We gratefully acknowledge the financial support granted by the Center for Information Technology (Center for Software Process Improvement) Denmark and the Danish Computer Technology Forum.

References

[1] Computer World Denmark (1999), Vol. 19, No. 82 & 88, December 1999.

[2] Zahran, S. (1997). Software Process Improvement - Practical Guidelines for Business Success. Addison Wesley Longman, Harlow, UK.

[3] Thomsen, H.E., P. Mayhew (1998). Approaches to Software Process Improvement. In Software Process - Improvement and Practice, Vol. 3, Issue 1, pp. 3-17.

[4] Jonassen Hass, A. M., et al. (1997), Bootstrap – the real way to SPI, Quality Week Europe 1997, quoted from Center for Software Process Improvement (2000), Delta report D- 263, pp. 17- 34, Hørsholm, Denmark.

[5] Center for Software Process Improvement, 1998, Danish Experiences with the Improvement of the Software Process, (in Danish). DELTA report D-262, Hørsholm, Denmark.

[6] ISO9001 (1987). Quality Systems - Model for quality assurance in design, development, production, installation and servicing. European Standard EN29001, Brussels, Belgium.

[7] TickIT (1992). A guide to software quality management system construction and certification using EN29001, Issue 2.0. UK Department of Trade and Industry, London, UK.

[8] Humphrey, W. S. (1989). Managing the Software Process. Addison-Wesley, Reading, USA.

[9] Kuvaja, P., et al. (1994). Software Process Assessment & Improvement - The Bootstrap Approach. Blackwell, Oxford, UK.

[10] El Eman, K. et. al. (1997). SPICE - The Theory and Practice of Software Process Improvement and Capability Determination. IEEE Computer Society, Los Alamitos, Ca., USA.

[11] Basili, V. R., H.-D. Rombach (1988). The TAME Project: Towards Improvement - Oriented Software Environments. In IEEE Transaction of Software Engineering, Vol. 14, No. 6, pp. 758-773.

[12] Pulford, K. et al. (1996). A quantitative approach to Software Measurement - The Handbook. Addison-Wesley, Reading, USA.

[13] Andersen, E. B (1991), Statistics for civil engineers, (in Danish). Akademisk Forlag, Copenhagen, Denmark

[14] Andersen, I. (ed.) (1990) Choosing methods for organizational Sociology, (in Danish). Samfundslitteratur, Copenhagen, Denmark.

[15] Larsen, E. Å., K. Kautz (1997). Quality Assurance and Software Process Improvement in Norway. In Software Process - Improvement and Practice, Vol. 3, pp.71-86, 1997.

[16]European Observatory on Software Engineering (1995). Final Report, PAC France/Germany.

[17] Wang, Y. (1998). Analysis report of the Swedish Benchmarking Survey of Software Development Practices. IVF, Technical Report, Mölndal, Sweden.

[18] Paulk, M. C. et al. (1991). Capability Maturity Model for Software. Technical Report CMU/SEI-91-TR24. CMU/SEI Pittsburgh, Pen., USA.

[19] Lee, M. S. Dutta, L. v. Wassenhove (1999). A Empirical Analysis of Software Production Problems in European Software Units. In Pries-Heje, J. et al. (eds.), Proceedings of the 7th European Conference on Information Systems, Vol. II, pp. 465-481, Copenhagen Business School, Denmark.

[20] Herbsleb, J. et al. (1994). Benefits from CMM-based Software Process Improvement: Initial Results. Technical Report CMU/SEI-94-TR13, ESC, R-94-013. CMU/SEI Pittsburgh, Pen., USA.

[21] Kautz, K. et al. (2000). Applying a Software Process Improvement Model in Practice: The Use of the IDEAL Model in a Small Software Enterprise. In Proceedings of the International Conference of Software Engineering, 4-11, June, 2000, Limerick, Ireland

[22] Brodman, J. D., D. L. Johnson (1994). What Small Businesses and Small Organizations say about the CMM. In Proceedings of the 16th International Conference on Software Engineering, IEEE Computer Society, pp. 331-340, May 1994.

[23] Brodman, J. D., D. L. Johnson (1997). Tailoring the CMM for small businesses, small organizations and small projects. In Software Process Newsletter, IEEE Computer Society, No. 8, Winter 1997.

[24] K. Kautz (1998). Software Process Improvement in Very Small Enterprises? Does it pay off? In Journal of Software Process - Improvement and Practice, Special Issue on Organizational Change through Software Process Improvement, Vol. 4, No.4, pp. 209-226.

[25] Paulk, M. C. (1999). Using the Software CMM in Small Organizations. Technical Report CMU/SEI-99. CMU/SEI Pittsburgh, Pen., USA.

[26] Rogers E. M. (1983). Diffusion of Innovations, Third Edition. The Free Press, New York.

[27] Oakland, J. S. (1993). Total Quality Management (2nd edition). Butterworth-Heinemann, Oxford, UK.
[28] Fenton, N. E.., Pfleeger, S. L. (1997). Software Metrics - A Rigorous and Practical Approach, PWS Publishing Company.
[29] Pfleeger, S. L. et al (1997). Status Report on Software Measurement. In IEEE Software pp. 33-43, March/April 1997.
[30] Kautz, K. (1999). Making Sense of Measurements for Small Organizations. In IEEE Software, Vol. 16, No. 2, pp. 14-20.
[31] Iversen, J. H. , Mathiassen, L. (2000). Lessons from Implementing a Software Metrics Program. In Proceedings of HICSS 33, Wailea, Maui, Hawaii.
[32] Bang, S. et al.(1991), Quality Management in Systems Development, (in Danish). Teknisk Forlag, Copenhagen, Denmark.
[33] Borum, F. (1995). Strategies for Organizational Change (in Danish). CBS Publishing Company. Copenhagen, Denmark.
[34] Andersen, N. E., et al. (1990). Professional Systems Development - Experience, Ideas, Action. Prentice Hall, Hemel Hempstead, UK.

Session 11 - SPI and Procurement

Session Chair: Pekka Forselius, STTF

Software Acquisition: Experiences with Models and Methods

Gerhard Getto
Thomas Gantner
Ton Vullinghs
DaimlerChrysler, Research and Technology
Ulm, Germany

Abstract

At DaimlerChrysler, the greatest part of the business and automotive software is provided by suppliers, often developed in complex software acquisition projects. Therefore, the development of a mature software acquisition process is essential for business success. One task of the DaimlerChrysler software process research department is the evaluation, customisation, and implementation of methods and techniques for software acquisition. This paper gives an overview of software acquisition process improvement activities that were performed at DaimlerChrysler. We used two SPRITE-S^2 software acquisition initiatives (ASSIST and PULSE) as a source of information for our own generic software acquisition reference process (GARP). Furthermore, we describe how we used these initiatives as an implementation basis for our software acquisition assessment method (SAM).

Introduction

As in any other branch of industry, software is becoming an increasingly important factor for DaimlerChrysler's business success. Over the last years several large projects, as well in automotive as in non-automotive areas, have been carried out, in which the development of technically complex software applications played a primary role. DaimlerChrysler's core business is the development of transportation products and services. A large part of software development is being out-sourced. To fulfil its position as a premium quality producer, DaimlerChrysler has to select and manage its software suppliers in a professional way.

An important task of DaimlerChrysler's process engineering research group is to study, develop, and evaluate relevant methods for software acquisition. Normally, the results of these research activities are tailored and transferred to the business units. As a part of these research activities we have been working on the comparison of different SPRITE-S^2 software acquisition models. SPRITE-S^2 is an initiative of the European Commission to support and guide in the procurement of ICT systems and services conforming to open specifications, and to promote and disseminate best practice by the application, validation and demonstration of existing new methodologies and tools. In this paper we describe how the results of the ASSIST [1] and PULSE [11] projects were used as an input and as a frame of reference for our own generic software acquisition process and assessment method.

Goal of our activities is to establish a continuous improvement process for the software acquisition processes of our internal customers. Our assessment method and generic reference process (including best practices) are important inputs for such an improvement cycle (see Fig. GGV.1):

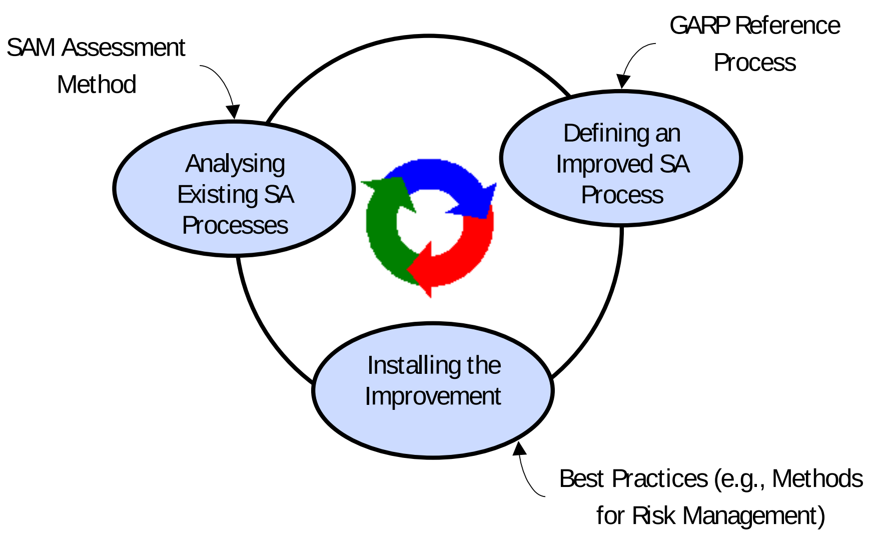

SAM Assessment Method
GARP Reference Process
Analysing Existing SA Processes
Defining an Improved SA Process
Installing the Improvement
Best Practices (e.g., Methods for Risk Management)

Fig. GGV.1: Software acquisition improvement cycle

To achieve our goal, we proceeded in the following way: First, a generic software acquisition reference process including a set of best practices was developed. This process is based on an internal DaimlerChrysler acquisition process enriched with elements of the Euromethod [2]. The generic reference process was compared to the acquisition processes as described by ASSIST, and PULSE. In parallel, we performed an ASSIST-based self-assessment at our marketing-, sales- and after-sales department, thus gaining first experience on software acquisition assessments. Furthermore, additional self-assessments in different departments of the passenger car division were performed, based on the PULSE method. These experiences led to the decision to take PULSE as a starting point for our own software acquisition assessment method. We have customised the PULSE assessment method to the special needs of our business units. Currently, we are working on the integration of the assessment method and the generic software acquisition reference process.

Overview. In Section 2 we describe our generic acquisition reference process (GARP) and relate this model to the SPRITE-S^2 approaches PULSE and ASSIST. In particular, we describe our criticism on these models and motivate the tailoring steps that we had to make. In Section 3 we describe the development of our software acquisition assessment method (SAM). Again, we relate the assessment method to the procedures as described in ASSIST and PULSE. Moreover, we briefly document our experiences with the application of the assessment model in our business units. Finally, in Section 4 we describe the integration of GARP and SAM.

Software Acquisition According to GARP

Motivation

Software acquisition and supplier management are a critical issue for many enterprises world-wide.

Software acquisition is concerned with the provision of an organisation’s need for software. Supplier management applies to management tasks while dealing with suppliers or subcontractors. Their impact on business processes is increasing dramatically. US enterprises spend more than $250 billion per year for procuring software products and services. Software investments may result in essential productivity and customer benefits as well as in strategic advantages. Following a systematic approach for software acquisition and supplier management might help to increase the benefits on these investments.

DaimlerChrysler is purchasing more and more software products. Software acquisition and supplier management are major issues. Professional management of a customer-supplier-relationship sets a number of prerequisites, such as requirement management, risk management, well-established communication, legal aspects as well as controlling ambitions and costs. To avoid failure of a software acquisition, a strategy for procuring software needs to be in place. The elements of such a software acquisition strategy should consist of defined processes based on best practices for each phase of the acquisition lifecycle.

There are numerous existing standards and emerging approaches for software acquisition world-wide. Most often they are conflicting and a general trend for uniformity is not perceptible. A DaimlerChrysler study on models and approaches for software acquisition and supplier management [5] gives an overview of the state of the art of key processes in this area. In this survey, models, methods and best practices were described, evaluated and compared. We used the results of this survey to build a generic software acquisition reference process (GARP) and a list of references to best practices for singular sub-processes. The foundation of this software acquisition reference process was the DaimlerChrysler IT project management handbook. This handbook describes project and quality management processes and techniques. We compared our company process to the following approaches:

- SA-CMM [12]: SEI’s capability maturity model for software acquisition.
- ISO15504 (SPICE) [9]: a major international initiative to support the development of an international standard for software process assessment.
- IEEE Std 1062 [8]: a consensus of broad expertise on software acquisition.
- ISPL [10]: an acquisition lifecycle oriented set of best practices for the management of ICT related acquisition processes .
- ASSIST [1]: a guide for the application of ICT procurement process improvement techniques.
- PULSE [11]: a methodology for assessment and benchmarking of ICT procurement.
- FAA-iCMM [3]: a capability maturity model that combines three existing maturity models for software development, system development and software acquisition, developed by the Federal Aviation Administration (FAA).

We detected a number of relevant sub-processes that were not covered by our company internal process. Accordingly, we extended the company internal process. The result is a comprehensive software acquisition reference process that may be used for the definition of a tailor-made business unit software acquisition process.

SPRITE-S^2 Approaches for GARP

In this section give a brief description of the two SPRITE-S^2 approaches we used as an input for our work: ASSIST and PULSE.

ASSIST deals with application of procurement process improvement techniques for small enterprises (SME). Some benefits provided are [1]:

- Receiving an ICT procurement improvement plan specific to their organisation needs with little investment in time and money.
- Gaining an understanding in a short period of time those procurement actions which have significant success rates and which are applicable to SMEs.
- Receiving a PC tool which is consistent with industry best practices to regularly measure and monitor the success of the procurement improvement actions.

PULSE provides a methodology for assessment and benchmarking of procurement processes. It is an organisational method for improving ICT procurement. Some main objectives of the methodology work of the PULSE project are [11]:

- Provide a means for organisations to assess their procurement practices using a consistent and reliable methodology.
- Establish a basis for an industry standard for procurement assessments based on extending the assessment specifications and production models currently established by the International Standards Organisation (ISO).
- Provide a tool which will enable organisations to identify and prioritise actions to improve their procurement processes.

ASSIST and PULSE follow two different strategies. ASSIST follows a bottom-up approach, that is, it helps to evaluate and improve software acquisition by defining a list of best practices and improvement recommendations. Its strength is to give hints for improvement. The goal of ASSIST is not to define and to structure a complete software acquisition process. Therefore the influence of ASSIST concerning GARP is located mainly on the level of best practices and not on the process level. We do not use the ideas of ASSIST for the evaluation and definition of software acquisition processes. Rather we use it as a source for potential improvements in our processes.

PULSE is a top-down oriented software acquisition assessment approach, that is, it uses the ideas and structure of ISO/IEC 15504 (SPICE) to define its assessment model. PULSE is the intended ISO standard for software acquisition processes. PULSE is a comprehensive description of relevant sub-processes for successful software acquisition projects and therefore an important input for GARP. We could not directly use PULSE for structuring GARP, because PULSE is a framework and not a lifecycle model. That means, PULSE does not define a logical ordering for all the contained sub-processes. The strength of PULSE is its comprehensiveness. We used PULSE as a kind of checklist for the sub-processes of GARP. Another way in which we actually use PULSE is for the assessment of existing acquisition processes (see Section 3). An important reason why we prefer PULSE above other approaches is its strong analogy with the assessment structure of the ISO 15504 standard (SPICE). Because one of our main internal customers is using SPICE oriented assessments for performing software supplier assessment, it is easier for us to communicate the main goals and underlying principles of PULSE to our customers. An issue that is still discussed is the way we can use the best practices as defined in PULSE. Regarding best practices PULSE breaks the structure of ISO 15504. PULSE defines best practices related to software acquisition also for level two and higher. This does not make sense according the philosophy of ISO 15504, because ISO 15504 only defines generic management practices for these levels.

The Generic Software Acquisition Reference Process (GARP)

The actual version of the developed generic software acquisition process is a hierarchical lifecycle model. From the perspective of our customers it was important to show the logical sequence of the sub-processes (if possible), as this ordering corresponds to already existing descriptions of other processes in place on the customer side and it supports the understandability of GARP. In this section we present an excerpt of this model in top-down order.

The first level of GARP contains the main processes like

- Acquisition initiation.
- For every single procurement: driving the procurement process.
- Review acquisition project

These processes are further refined. For example the process "Driving the Procurement Process" consists of the sequential sub-processes shown in Fig. GGV.2.

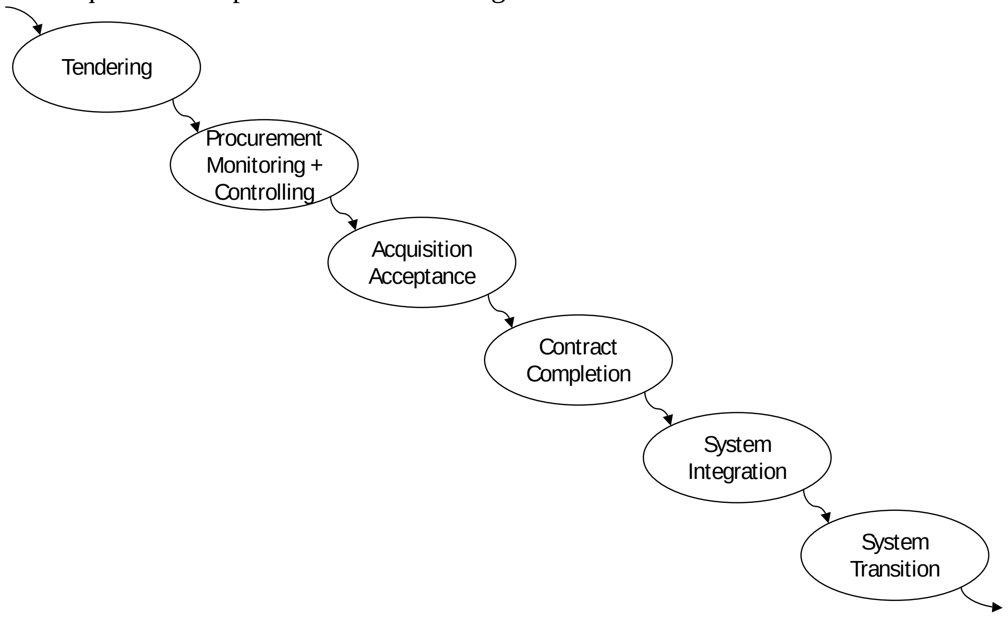


Fig. GGV.2: Driving the procurement process

On the next level the sub-process "Procurement Monitoring and Controlling" is divided into parallel and continuous sub-processes (see Fig. GGV.3). We did not describe any logical sequence here, as the represented sub-processes possess a strong parallel and continuous character.

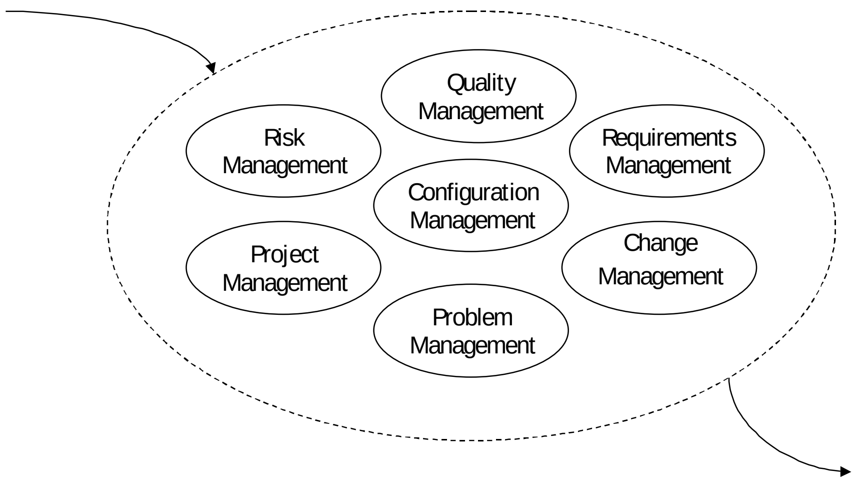


Fig. GGV.3: Procurement monitoring and controlling

On the lowest levels of GARP there are links from the sub-processes to best practices. The best practices provide methodological support for performing these sub-processes. The links represent the connection between *what* to do for software acquisition and *how* to perform the sub-processes. In one of our earlier research papers [7] we documented best practices for the risk management process. Fig.

GGV.4 summarizes the overall structure of the activities that lead to the definition of GARP.

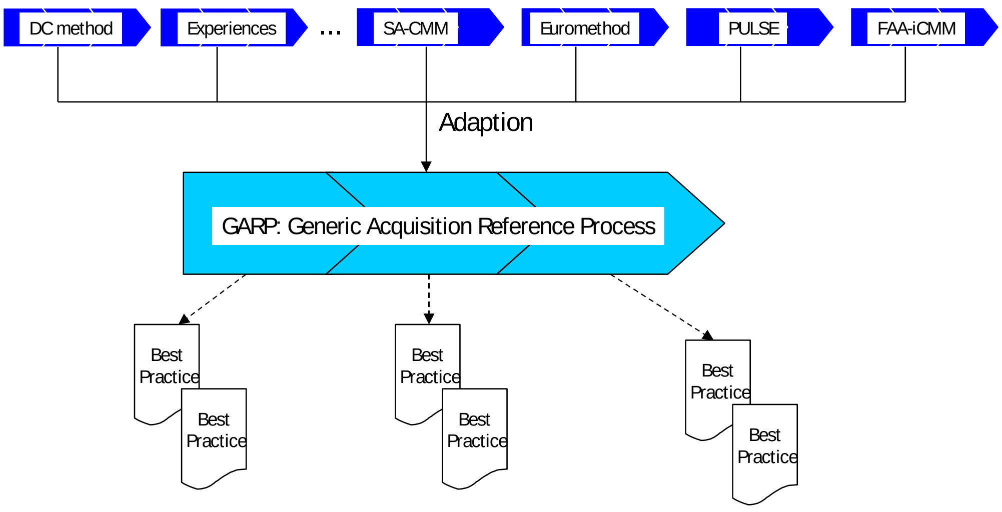


Fig. GGV.4: The construction of GARP

Best practices support the introduction of improvements. They are a source of possible improvements for existing acquisition sub-processes. Of course, finding and defining best practices is an ongoing activity. The strategy for building a set of best practices that support our customers is described at the end of this paper.

Software Acquisition Assessments

Motivation

Within a process improvement context, process assessment provides the means of characterising the current practice within an organisational unit in terms of the capability of the selected processes. Analysis of the results in the light of the organisation's business needs identifies strengths, weakness and risks inherent in the processes. This, in turn, leads to the ability to determine whether the processes are effective in achieving their goals, and to identify significant causes of poor quality, or overruns in time or cost. These provide the drivers for prioritising improvements to processes. (SPICE Introduction [9])

DaimlerChrysler is getting more and more aware of the importance of software acquisition processes. In 1998, first experience with software acquisition processes were collected in the passenger car development business unit (PCD). SEI's software acquisition capability maturity model (SA-CMM) [12] was used as a reference model for process improvement. The key process areas that we focussed on were requirements management, evaluation of intermediary and final products, solicitation, and risk management [4]. The reasons to select those areas were mainly driven by our own experiences in those fields and ad-hoc decisions of our customers. In particular, we did not perform assessments to identify strengths and weaknesses in a structured way. To bridge this gap and to better respond to the needs of our customers, we decided to develop our assessment method. Since GARP was still under construction by then, there was no assessment method available that uses GARP as a reference process model. We decided to develop a DaimlerChrysler specific software acquisition method based on already existing approaches ASSIST and PULSE.

The following steps led to the definition of our software acquisition assessment method (SAM):

1. collect assessment experiences using ASSIST and PULSE
2. select a basic software acquisition method
3. customise this method to DaimlerChrysler specific constraints
4. collect experiences by applying the customised method in pilot projects
5. improve the customised method based on these experiences
6. integrate SAM and GARP

Experiences using ASSIST and PULSE

The first customer for software acquisition assessment and improvement was the IT Marketing, Sales, and Aftersales Department, which accelerates the development, deployment, and support of IT solutions.

When planning the assessment process, the presence of experienced project managers is crucial. PULSE is much more comprehensive and therefore more time-consuming in its execution than ASSIST is. Because we wanted experienced project managers to participate in the assessment, we decided to use the ASSIST assessment. The assessment process and results are described in one of our experience reports [6].

First experiences with PULSE were gained in a large project in the passenger car development business unit (PCD). A special software quality taskforce was interviewed in a self-assessment workshop to analyse the current project situation, to solve existing problems, and to find pointers to areas for process improvement. The assessment results helped us to extend an already existing activity list with new, urgent activities.

The experiences affirmed our impressions that ASSIST is a pragmatic, best practice based bottom-up approach, whereas PULSE is a comprehensive, theory and best practice based top-down approach. When shortage of time of manager, project leader and project members is the biggest assessment constraint, ASSIST or similar approaches could be used to perform a “light assessment”. If the assessment starts a systematic process improvement activity, PULSE is, in our opinion, a better alternative.

Customizing a selected method to DaimlerChrysler specific constraints

Because of its systematic, comprehensive, and SPICE oriented model, we selected PULSE as a basis for SAM. Based on experiences with ASSIST and PULSE we got process descriptions and supporting practices for two different assessment types:

- A “light assessment”, which comprises a well prepared assessment workshop
- A “full assessment”, which consists of the whole PULSE assessment process.

Some of our customer wanted to use the software acquisition assessment as a starting point for an extensive improvement program. In this case, a “light assessment” would not return enough useful information. On the other side, a “full assessment” requires a lot of resources and is only applicable in an organisation if the improvement initiative is initiated by upper management. This was not the case in our company. We decided to develop a “restricted assessment process”. It is a restriction of the PULSE assessment and is based on interviews and workshops. The main differences between SAM and the PULSE assessment method are:

- SAM is restricted to the PULSE capability levels 1 – 3,
- SAM omits some of PULSE process steps (e.g., “determine assessment reference model, assessment model and tool”),

- SAM combines several PULSE process steps into one step,
- SAM adds extra agreement steps and SAM adds an improvement planning step.

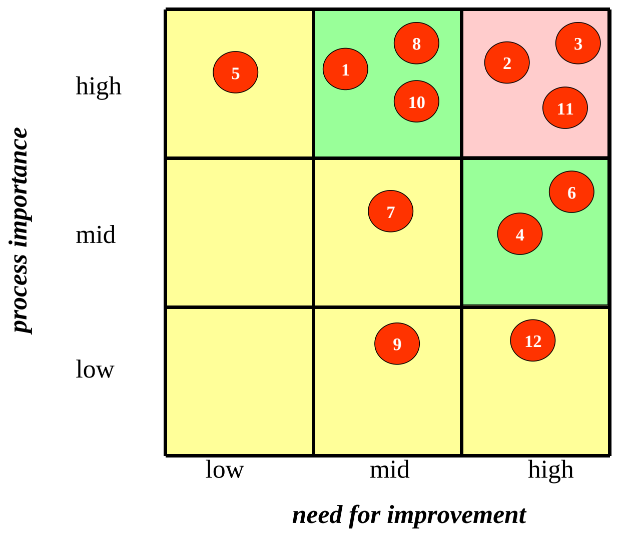
Our actual restricted assessment process contains ten steps:

1. *Presentation of assessment approach to managers:* A workshop will be held with participation of all managers of departments that will be assessed. Goal is to introduce the assessment method, get a commitment to do it, select projects and people for the actual assessment, and define a first rough assessment schedule.
2. *Identification of existing processes:* A structured interview with a representative of each organisation or project who participates in the assessment is performed. One goal is to describe existing processes and/or practices and roughly classify them according to related PULSE process categories. Additionally, the interviewed people give their opinion on which processes should be assessed.
3. *Presentation of assessment approach to the assessment team:* A workshop is held with all people who participate in the assessment. The goals are to introduce the assessment method, present the result of the interviews (step 2), select processes that have to be assessed, identify constraints for the assessment, and define a detailed assessment schedule.
4. *Inventory of selected projects:* During the interviews in step 2 people are asked to give additional information about their project. The documents are analysed by the assessors in order to get a list of used processes, standards, techniques and best practices.
5. *Assessment interviews:* Department managers, project leaders, project participants and method specialist are interviewed to disclose the strengths and weaknesses of the selected projects.
6. *Analysis of interview results:* The assessors analyse the results of the interviews (step 5). The goals are to compile the strengths and weaknesses results of all interviews, to derive ratings and capability profiles, and to discover contradictions.
7. *Assessment workshop:* A third workshop is held with all people who participate in the assessment. The results of step 6 are presented and discussed, especially the contradictions. The goals are to get an agreement on process strengths and weaknesses, to present ratings and capability profiles, and to select and agree in a prioritised list of processes that should be improved.
8. *Improvement planning:* The assessors develop a proposal for an improvement plan regarding the most important processes. It contains a list of activities, resources, and a schedule.
9. *Presentation of improvement plan to assessment team:* A final workshop is organised with all people who participate in the assessment. The improvement plan is presented and discussed. Goal is to get an agreement on this plan.
10. *Presentation of improvement plan to management team:* The assessment results including the improvement plan is presented to the top management of the assessed organisation. They have to agree in the proposed improvement activities and release the necessary resources. A second goal is to get a commitment for a continuous software acquisition assessment and improvement program.

Currently, we are performing a “restricted assessment” at the marketing, sales and aftersales department. Goal of the assessment is not only to find pointers to areas of improvement for this department, but also to improve our own assessment method. The overall amount of time 16 participants have to spend is approx. 30 days. Preliminary results justify our structured medium-weight approach.

Besides gentle side-effects like creating a better awareness for process improvement issues, the restricted assessment leads to a more transparent and understandable selection of improvement areas. For example, to support the selection of the processes that have to be assessed (step 3), we used a simple portfolio technique to rank the different processes (see Fig GGV.5). We created a similar portfolio after every individual assessment interview (step 5). By comparing these portfolios we could verify whether individual answers conflict with the results of the second workshop (step 3). Conflicts will be discussed and resolved in the third workshop (step 7). Furthermore, the individual portfolios clearly illustrate which participants are experts regarding some specific process. In this way, the exchange of best practices among all participants is promoted.

Fig. GGV.5: Process prioritisation using a portfolio technique (the numbered bullets refer to different software acquisition processes).



Integrating GARP and SAM

To have an effective overall software acquisition methodology, the scope of processes that can be evaluated by our assessment method should correspond to the sub-processes of GARP and vice versa. As mentioned before, GARP is based on an extended DC-internal life-cycle model for software acquisition. On the other hand, our assessment method is strongly related to the acquisition assessment model of PULSE. The question arises how to integrate these two models. In Fig. GGV.6 we have depicted the mapping that we have to perform for this integration.

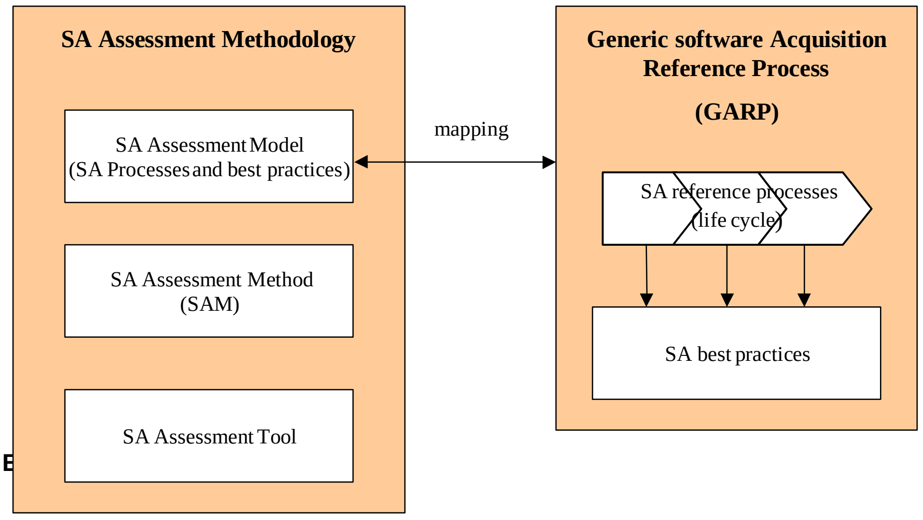

SA Assessment Methodology
SA Assessment Model (SA Processes and best practices)
SA Assessment Method (SAM)
SA Assessment Tool
mapping
Generic software Acquisition Reference Process
(GARP)
SA reference processes (life cycle)
SA best practices

Fig. GGV.6: Integrating GARP and SAM

In accordance with the PULSE methodology, our overall assessment methodology contains three main components:

- the software acquisition assessment model, describing acquisition processes and relating them to capability levels by using process attributes and best practices. In our assessments we basically used the PULSE model. Goal of the integration is to construct a model that is compatible with our generic reference process (GARP).
- an SA assessment method, describing *how* to use the assessment model (eventually in combination with an assessment tool). Our SA assessment method was described in the previous section (SAM).
- a tool to collect the assessment data and to perform the calculations and graphing of the resulting profiles. Currently, we do not use a tool in our assessments.

Since GARP partially originates from PULSE, it is possible to list for every (for our company relevant) PULSE process a corresponding GARP process. On the other hand, GARP contains processes that are not covered by PULSE. For these processes we have to adapt the assessment model. For example, GARP contains a "transition to support" process, describing how to prepare and perform the transition of a developed system in its operating environment. For this process we have to define capability levels, process attributes, and best practices.

The actual state of the integration is that we have identified those processes in GARP that are not yet covered by the PULSE processes. We are currently working on the theoretical extension of the acquisition model. In parallel, we try to extend our set of best practices based on real-life experiences in software acquisition process improvement.

Conclusions

We presented our research activities in the field of software acquisition process improvement. We described a generic acquisition reference process that guides the improvement of existing acquisition processes. Pointers to areas for improvement are delivered by process assessments. We developed a software acquisition assessment approach based on the PULSE methodology. Finally, we sketched how we will integrate our generic process and assessment method.

As the integration activities are not finished yet, this will be a main topic for further research. Another future activity will be the application of the developed concepts in co-operation with our company-internal customers and the recording of these experiences. First assessments show promising results: process evaluations are performed in a well-structured and efficient manner. Especially the latter is important for our customers. Another necessary task is to extend our generic acquisition reference process with more best practices, supporting methods and facilities. The long term goal is to provide complete methodological support for our generic reference process. This means that all relevant sub-processes are covered from the process side as well as from the methodical support side.

References

[1] ASSIST: *Leitfaden für die Beschaffung von IT für mittelständige Unternehmen*, QAI Europe, 1999.
See also: http://www.ispo.cec.be/sprites2/fi-assist.html

[2] Euromethod: http://www.ispo.cec.be/sprites2/spriemet.htm

[3] Federal Aviation Administration: *Integrated Capability Maturity Model*, http://www.faa.gov/aio/icmm/FAA.htm

[4] Gantner T., Weber T.: *A Way to a Comprehensive Process Model for Software Acquisition from the Viewpoint of the Customer*, Proceedings of the 6th European Conference on Software Quality, 1999

[5] Getto, G.: *Evaluation von Prozessen und Methoden für Softwarebeschaffung und Management von Auftragnehmern*, Proceedings of the 4th Kongress Software Qualitätsmanagement “Made in Germany”, 1999

[6] Getto, G., Gantner T.: *Software Acquisition Processes at DaimlerChrysler AG: Research Activities for Improvement*, Proceedings of the European Conference on Software Process Improvement (SPI99), 1999

[7] Getto, G., Landes, D.: *Risk Management in complex Project Organizations: A Godfather-driven Approach*, Proceedings of the 30th Projektmanagement Institute Conference (PMI99), 1999

[8] IEEE 1062: *IEEE Recommended Practice for Software Acquisition*, IEEE Standards Association, 1993

[9] ISO/IEC 15504: *Information Technology – Software Process Assessment*, ISO/IEC JTC1/SC7/WG10, 1998

[10] ISPL: *ISPL (Information Service Procurement Library) Managing Acquisition Processes*, ten Hagen & Stam, 1999

[11] PULSE: *A Methodology for Assesssment and Benchmarking of Procurement Processes*, http://www.ispo.cec.be/sprites2/fi-pulse.htm

[12] SEI: *Software Acquisition Capability Maturity Model (SA-CMM) version 1.01*, Technical Report, CMU/SEI-96-TR-020, 1996

About the Authors

Thomas Gantner studied mathematics at the University of Ulm in Germany. After obtaining a master’s degree in 1991 he joined the software process engineering group at the DaimlerChrysler Research Laboratory. Here he is employed as a research manager. He is performing research and consulting activities in the areas of software quality management and software process design and improvement, especially for software acquisition processes. At the moment he is the project leader of a long term research project with the passenger car development business unit as the key customer. Other company-internal customers he worked with are in the field of technical and administrative software. Gantner can be reached at DaimlerChrysler AG, P.O. Box 23 60, D-89013 Ulm, Germany, or by e-mail at thomas.gantner@DaimlerChrysler.com.

Gerhard Getto is a manager at DaimlerChrysler Research and Technology in Ulm, Germany. His research activities concern IT management processes and methods like risk management, quality management, software acquisition management, and supplier management. Getto has a degree in mathematics from the University of Stuttgart, Germany. He has 19 years of experience in the IT area. During this time he was active in software development, system implementation world-wide, business process analysis, information management, project management, quality management, risk management, and research. He has published papers in the areas of risk management, quality management, and software acquisition. Getto can be reached at DaimlerChrysler AG, P.O. Box 23 60, D-89013 Ulm, Germany, or by e-mail at gerhard.getto@DaimlerChrysler.com.

Ton Vullinghs studied computer science at the Catholic University of Nijmegen in the Netherlands. After obtaining a master’s degree in computer science in 1993 he joined the software methodology and compiler construction group at the computer science department at the University of Ulm (Germany). Here he was employed as a teaching and research assistant and obtained a PhD for his research in the field of functional programming and graphical user interfaces. Currently, Ton Vullinghs is working as a research assistant at the DaimlerChrysler Research Laboratory in Ulm. He is performing research and consulting activities in the areas of software quality management and software process design. His main topics of interest are software risk management and software acquisition management. He can be reached at DaimlerChrysler AG, P.O. Box 23 60, D-89013 Ulm, Germany, or by e-mail at ton.vullinghs@DaimlerChrysler.com.

Company Description

DaimlerChrysler AG is a stock company organised under the laws of the Federal Republic of Germany. DaimlerChrysler provides a wide range of transportation products and financial and other services. DaimlerChrysler is world renowned for its high quality products which reflect a long tradition of exceptional engineering, performance, service and safety. DaimlerChrysler’s Research and Technology Division is the hub of activity when it comes to the securing of the Group´s technological future. Its responsibilities are to support the business units in the development of their technology strategies, to ensure integrated innovation and technology management and to create the technological basis for differentiating products in the marketplace. The software process engineering research group is a competence centre for software processes and possesses, among other competencies, theoretical and practical know-how in the field of software acquisition management. One of the main objectives of the research group is to tailor and transfer relevant results to the business units of DaimlerChrysler.

Rational Unified Process in a public procurement environment

Annette Ibsen
TietoEnator Consulting A/S, Denmark
Jesper Rugård Jensen
TietoEnator Consulting A/S, Denmark

Abstract

Rational Unified Process has by now gained widespread acceptance. The process espoused by Rational Software Corporation supports modern object oriented iterative/incremental development focusing on requirements and architecture. There are, however, problems applying this type of development process. Using the process in a public procurement environment is hard as there are gaps between what the process recommends and what is actually possible. There are many challenges in deploying the Rational Unified Process, but the main problem is that the Rational Unified Process recommends that plans and budgets are finalised relatively late in the process and it is recommended that plans and requirements are adjusted throughout an iterative process.

In a public procurement environment (or rather in most situations of competitive tendering) there is a clear signoff between the offer and the tender. Most often the offer is fixed and any changes requires new contracts and possibly a new procurement process. This is in contradiction to the Rational Unified Process where the requirements are being adjusted throughout the process and furthermore are tested by the parallel implementation of an executable architecture.

In this way the iterative incremental process is diluted (and parts of the analysis activities are removed from the projects). There is thus a risk of the losing the gains of: better control of time, budget and quality.

It is our intention to discuss how to solve this focusing both on the long run and what to do in the short run, i.e. how to adapt the Rational Unified Process to a world of competitive tendering.

Introduction

On the following morning there was a great noise of trumpets and drums, and a procession passed through the town, at the head of which rode the King's son. Behind him came a herald, bearing a velvet cushion, upon which rested a little glass slipper. The herald blew a blast upon the trumpet, and then read a proclamation saying that the King's son would wed any lady in the land who could fit the slipper upon her foot, if she could produce another to match it. Of course, the sisters tried to squeeze their feet into the slipper, but it was of no use--they were much too large. [1]

One size fits all?

One glass slipper size for everyone? Not if you want to marry the prince and live happily ever after. Trying out other peoples glass slippers is likely to make you loose a heel or a toe.

Being hurt badly once by the hope of fitting into another's slipper, one isn't likely to try doing it again. This even though the shoe is touted as being flexible and a sure fit for nearly every one.

One the surface processes and glass slippers may not seem to have a lot in common, but the problems of fitting into them are very similar and the consequences of a misfit can be equally disastrous. Both physically and financially.

In the last couple of years the use of a process to support software development activities has gained widespread acceptance. One common denominator in these processes is that they are iterative/incremental. Rational Unified Process (RUP) is one of the new processes. One that right now is looking as *the* standard modern development process.

RUP is described as a flexible customisable process framework that should be instantiated before being used in a specific environment.

> The Rational Unified Process™ (RUP) is a Web-enabled software engineering process that enhances team productivity and delivers software best practices to all team members. RUP is a customisable framework, which can easily be adapted to work the way you work. [2]

RUP is described as a process capturing best practices in modern software development, making it suitable for a wide range of projects and organisations [3]. The limits of Rational Unified Process are not explicitly scoped in the literature (or at least not where we could find it), but the underlying impression, is that RUP fits all but the strangest of organisations and problem domains.

But does RUP fit all?

And how flexible is it?

Cinderella's stepsisters wouldn't likely buy a one size fits all process. They already got hurt once by their stepsister's shoe design.

In our daily work using RUP and adapting this process, we have met difficulties applying RUP in working with large organisations who are used to doing waterfall and fixed price projects. They have a hard time incorporating the concepts behind RUP and even though they often find the process interesting, they are too big, too busy or otherwise too set to effectively change direction overnight. What we have noticed is that when we succeed in adapting RUP to a company, they tend to be satisfied with the results.

This paper concentrates on the practical side of using RUP in these environments, but it could be about any modern iterative process such as OPEN [4], XP [5], DSDM [6] i.e. The questions spring from using the process (and parts of it) in practice and from observing our colleagues and customers trying to understand, adapt, and customise the process. One of the obvious stumbling blocks when using (or teaching/discussing) the process is the dichotomy between the iterative/incremental aspects of the process and the use of the process in large projects in a supplier/customer relationship, where a procurement process has laid the foundation for the project.

We want to explore where we stumble in our work with iterative, architecture-centric and use case driven processes and what to do to ameliorate these difficulties.

This paper consists of a short presentation of an iterative/incremental process exemplified by RUP, a description of one type of problems we have met in practice working with our customers and finally some recommendations for long and the short strategies using RUP.

These recommendations are not in any way a silver bullet, but more a way of easing the pain of adapting a new development process and as a starting point for further discussion.

Maybe the slipper can be eased on slowly.

A small note on technical terms

In this paper we talk about using RUP in an environment, where there is a customer/supplier relation expressed in some sort of contract describing the solution proposed by the supplier. This contract is normally based on a requirements specification made by the customer and any changes to the solution or the requirements normally results in a renegotiation of the contract.

Furthermore there is a split between who do the requirements analysis and who do the actual implementation of the system. This way of working is referred to as fixed price projects, a tendering process, a (public) procurement process. The differences between the above mentioned is mostly a matter of degrees.

Rational Unified Process

Here follows a short introduction to RUP. For a more detailed description see [2], [3] or RUP on-line documentation [4].

We focus on the parts of RUP relevant for the discussion of using RUP in a public procurement environment and will not try to cover all aspects of the process.

Rational has identified six best practices for developing software [3]:

1. Develop software iteratively
2. Manage requirements
3. Use component based architectures
4. Visually model software
5. Verify software quality
6. Control changes to software

For the purpose of this paper we will look closer at 1, 2 and 3 as they are closely linked to the problems of using RUP in a public procurement environment. Of the rest both 5 and 6 also has impact on the problems, but we will limit ourselves to the above mentioned for reasons of space.

Furthermore we will present the four fundamental phases of RUP as they are important for the discussion on contracts and how to apply RUP in a fixed price project.

Iterative development

Iterative development can seen as the possibility of redoing work, i.e. it is possible to return to doing analysis/design after doing implementation, if new information about the product is produced.

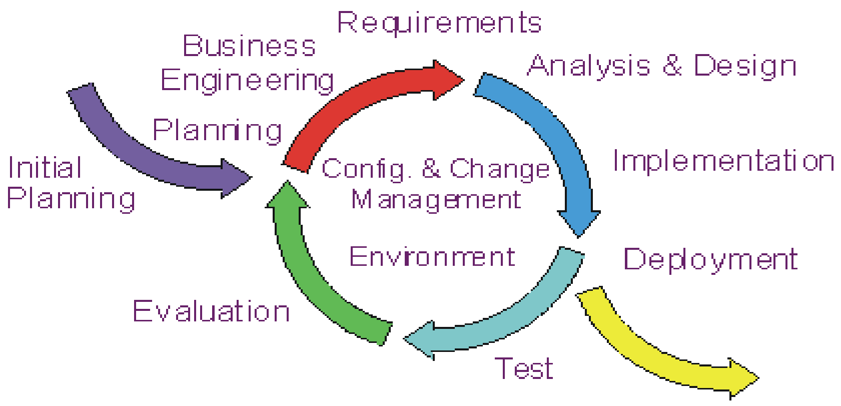


Fig. 1 Iterative Development

All software projects of some size are risky. There are unknowns in regards to requirements and technical matters. As software development is concerned with some kind of innovation it is not possible to predict all risks regardless of how experienced the development team is.

The earlier in the lifecycle you can verify that you have avoided a risk, the more stable the project is likely to be. The earlier a risk is handled the less it is going to influence large parts of the project in a negative manner.

In a waterfall lifecycle, it cannot be verified that risks have been avoided or not until late in the lifecycle and this could result in costly over-runs and in some cases even project cancellation. Because using a traditional development process many risks are not even discovered until it is attempted to integrate the different modules of the project.

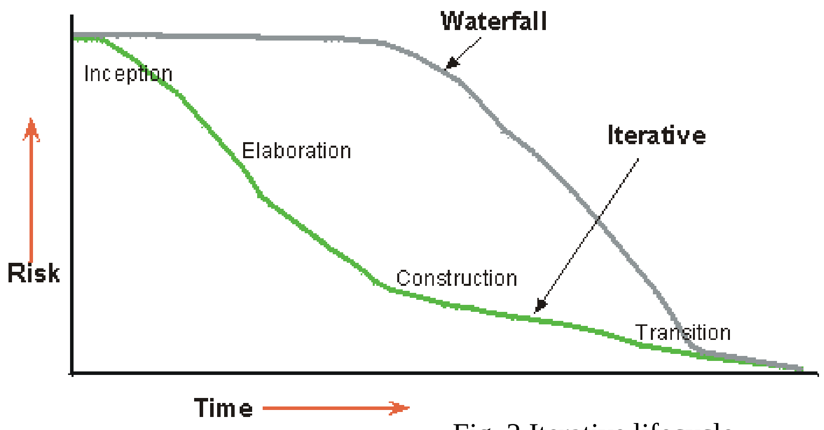


Fig. 2 Iterative lifecycle

In an iterative lifecycle, the iterations are planned in order to address the identified key risks in descending order of importance. Since the iteration produces a tested executable, it is possible to verify whether a targeted risk has been mitigated or not.

Some benefits of an iterative approach

Easier to change requirements - It is not possible to know everything at the onset of the project. Reality is that requirements change during a project's life and this have always been a problem.

Integration is not one "big bang" at the end because the project now is broken into smaller increments with integration earlier in the process and with fewer elements in the beginning.

Iterative development focuses on the architecture first, and in that way verifying the architecture by doing integration and allowing design flaws to be detected and resolved earlier in the lifecycle

Risks are mitigated earlier. The early iterations are the iterations where we are to discover and address the risks. That means we can resolve major risks before making large investments.

Manage Requirements

Management of requirements is about eliciting, organising, and documenting the requirements of the system to be developed, and establishing and maintaining agreement between the customer and the supplier when the requirements of the system changes.

Requirements are dynamic - they will change during the project life-cycle. Therefore you have to evaluate changes, determine their impact on the project and document the tradeoffs and decisions. Furthermore these changes will influence the architecture and the design of the system.

Component-Based Architecture

A software system’s architecture is perhaps the most important aspect that can be used to control the iterative and incremental development of a system throughout its lifecycle.

The early iterations of the process should focus on producing and validating a software architecture. This takes the form of an executable architectural prototype that should evolve gradually into the final system in later iterations. An executable architecture is a *partial* implementation of the system, built to demonstrate selected system functions and properties, in particular those satisfying non-functional requirements, but also the key requirements. It is built to mitigate risks related to performance, throughput, capacity, reliability etc., so that the complete functional capability of the system may be added in the construction phase on a solid foundation, without fear of breakage.

An example of an architectural concern is an integration risk. For example, does the chosen database work properly with the chosen operating system? Another example is performance risk. Is the chosen database fast enough?

Phases

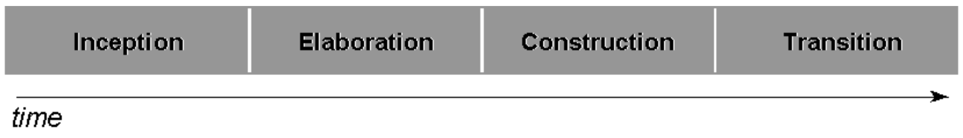


Fig. 3 The four phases in Rational Unified Process

RUP is composed of four sequential phases. An assessment is performed at each phase-end to determine whether the stated objectives of the phase have been met. A satisfactory assessment is necessary for the project to move to the next phase.

Inception

In the inception phase the scope of the project is defined. The main goal of the inception phase is to achieve agreement among the stakeholders on the lifecycle objectives for the project and the success criteria. The inception phase is of significance primarily for new development efforts, in which there are significant business and requirements risks which must be addressed before the project can be initiated.

The primary objectives of the inception phase include:

Exhibiting, and maybe demonstrating, at least one candidate architecture against some of the primary scenarios

Estimating the overall cost and schedule for the entire project (and more detailed estimates for the elaboration phase that will immediately follow)

Estimating potential risks (the sources of unpredictability)

Elaboration

The goal of the elaboration phase is to baseline the architecture of the system. This baselining is done in order to provide a stable basis for the design and implementation effort in the construction phase. The architecture evolves out of a consideration of the most significant requirements (both functional and non-functional) and a risk assessment. The stability of the architecture is evaluated through architectural prototypes.

The primary objectives of the elaboration phase include:

To ensure that the architecture, requirements and plans are stable enough, and the risks sufficiently mitigated to be able to predictably determine the cost and schedule for the completion of the development.

To address all architecturally significant risks of the project

To establish a baselined architecture derived from addressing the architecturally significant scenarios, which typically expose the top technical risks of the project.

It is first at the end of the elaboration phase that it is possible to answer the feasibility question in sufficient detail:

"Are the use cases, architecture and plans stable enough, and are the risks under sufficient control to be able to commit the whole development work in a contract?" [7]

Construction

The goal of the construction phase is on clarifying the remaining requirements and completing the development of the system based upon the baselined architecture. This phase focuses primarily on the implementation of the system and is meant to be a fairly stable phase as most of the risks hopefully have been resolved. It is still iterative though and changes to requirements can still arise.

The primary objectives of the construction phase include:

Completing the analysis, design, development and testing of all required functionality.

To iteratively and incrementally develop a complete product that is ready to transition to its user community.

To achieve some degree of parallelism in the work of development teams. A robust architecture is essential if any significant parallelism is to be achieved.

Transition

The focus of the Transition Phase is to ensure that software is available for its end users. The Transition Phase can span several iterations, and includes testing the product in preparation for release, and making minor adjustments based on user feedback.

The primary objectives of the Transition Phase are:

training of users and maintainers

assessment of the deployment baselines against the complete vision and the acceptance criteria for the product

achieving user self-supportability

Challenges in handling fixed price contracts while using the Rational Unified Process

When working with an iterative process like RUP and at the same time having to negotiate a contract very early in the project lifecycle, the supplier often risk losing the possibility to work iteratively in the first two phases (inception and elaboration). The supplier also risk losing the opportunity to build an architectural baseline and address critical risks etc. This is major risk for the project according to RUP and what is recommended in order to deliver quality on time and on budget.

In this part we would like to introduce a standard contract template (K33) used extensively in Denmark, the added challenge of participating in a EU setting and how the contract situation fits an iterative process like RUP. These examples should be seen as specific instances of what we see as general problems using an iterative process in an fixed prices setting.

Contract type K33 and EU setting

In Denmark there is one predominant type of standard contract used in public procurement scenarios:

Development of a new application (K33) [8]

The K33 covers deliveries that typically are the result of a lengthy development project and where installation and support often are included in the project. It is hard to formulate exact criteria for evaluating a K33 type of project because it often includes major new functionality.

The contract template supports sequential deliveries of modules where the customer accepts or declines the result on the basis of acceptance tests of the individual modules.

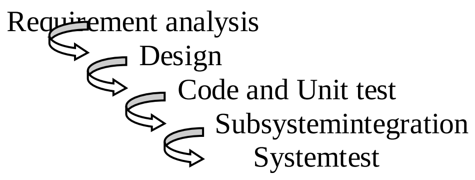
The activities defined by K33 are:

- Analysis (optional) – the supplier analyse the customers requirement specification.

- Proposal for solution – the supplier delivers a proposal for the solution based on either the requirement specification or on the analysis.
- Writing the contract.
- The supplier develops and deploys the system.

Any significant changes of the requirements or new constraints on the project lead to a new procurement process taking place.

The activities in the K33 contract maps to the waterfall model:



Waterfall is conceptually straightforward because it produces a single deliverable. The fundamental problem of this approach is that it pushes risk forward in time, where it's costly to undo mistakes from earlier phases

Looking at the K33 standard contract from the waterfall perspective:

The supplier analyse the specification → Requirement analysis
The supplier deliver a proposal for solution → Design
The supplier implements the system → Code and Unit test
The supplier deliver the system → Systemtest and deployment

In the waterfall model the requirements specification is produced in the requirement analysis phase. In the contract situation the customer delivers a complete specification and the supplier has to interpret that specification through e.g. use case modelling. I.e. it is the customer who does the requirement analysis and there is no provisions for the work that the supplier has to do in order to understand the specifications. This is covered in the contract, but the contract doesn't cover the requirements analysis. This split between who produces the requirements and who builds the solution is crucial to the problems with the use of RUP in a fixed price environment.

If the project furthermore is in a EU setting the time to produce the solution is limited. The steps in EU settings are [9]:

1) The tender material in the form of a requirements specification is sent to the suppliers chosen to participate in the tendering process.
2) At the latest **six weeks** after the tender has been received the supplier gives an offer based on a K33 contract template
3) The customer evaluates the contract and chooses a supplier and the contract is finalised

Normally, in relation to RUP, the customer produces a general list of requirement based on their business visions. In the elaboration phase the customer detail and interpret the requirements *together with the supplier* and at the same time the supplier builds the architecture and addresses most of non-technical critical risks. At the end of the elaboration phase both a complete requirement specification and a stable architecture are present and they form the basis for implementation and installation.

The typical project profile below shows the relative duration and effort per phase [3]. It is suitable for projects that has following characteristics:

- Is of moderate size and effort
- Is in an initial development cycle

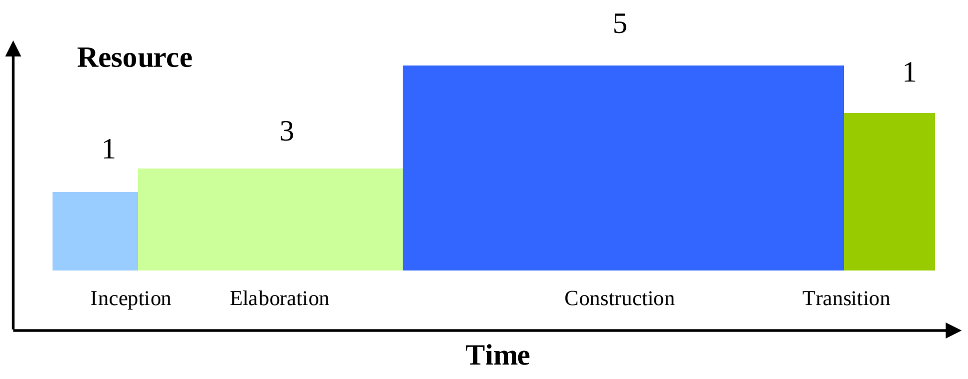


Fig.4 Typical project profile

When we look at a fixed price contract situation the customer will have completed the requirements specification and expects the supplier to be able to give an offer based on this specification, i.e. the customer expects to be able to move into the construction phase. The customer has completed both the inception and elaboration phases before the supplier enters the stage, but hasn't built an architectural baseline, which is a major critical risk according to RUP.

For the supplier, it will be necessary to reinterpret the requirements, in order to be able to give an offer for the project and therefore work still has to be done in what amounts to the elaboration phase, but time is normally limited for this as the customer considers the project to be ready to enter the construction phase and is not ready to pay for more work in the elaboration phase.

The time spent by the supplier doing this, is often very short and unpaid and it is not feasible to baseline the architecture and address risks. Because of this the elaboration phase gets compressed as illustrated in the figure below and the project may continue into the construction phase with major risks not mitigated.

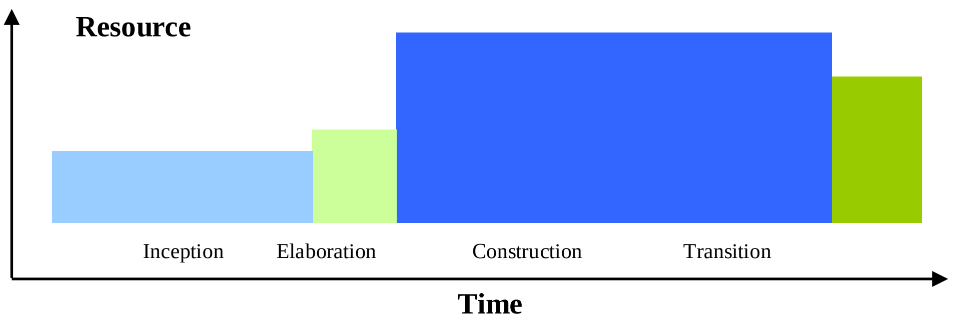


Fig. 5 Compressed project profile

In a tendering process it is not possible to reiterate and the supplier thus loses the opportunity to do iterative development in a meaningful manner.

The contract situation and Rational Unified Process

As mentioned earlier one of the main issues in RUP is to develop iteratively. Some of the gains we expect from doing iterative development are:

- Early handling of critical risks
- Early architecture baselining and thereby eliminating technical risks
- Early user feed back
- Early integration and deployment of increments
- Requirements management in a dynamic world

When the supplier is forced to give an offer for the project very early (often right after the inception phase) there is no opportunity for addressing critical risks, especially those that are concerned with resolving architectural risks.

The supplier doesn’t have the opportunity to baseline the architecture and address risk before the construction phase and that will be reflected in the eventual offer for the project.

> "A fixed price-contract for something that is not well understood either includes large safety factors or is an enormous gamble" [10]

This is in contrast to the development process described above and it necessitates some work in order to adapt RUP to these settings.

Strategies for doing iterative development in a public procurement environment.

We have identified two main strategies for dealing with the problem of doing iterative development in a fixed price environment. One is short term for dealing with the problems right now that doesn't force the customer into any big overnight changes of the development organisation. The second is a fundamental and long term solution focusing on the organisation culture, making reality fit the ideal.

Short term - Stopgap measures

Today, there is a tradition of making fixed price tenders based on an often very detailed requirements specification. And the life cycle of the project is mostly divided into two very different project stages: An engineering stage covering the inception and elaboration phases, which is done on a time/material basis and a production stage covering the construction and transition phases done fixed price specified in a contract. The engineering stage focuses on producing a complete requirements specification in order to control the rest of the project. Consequently there is no base lined architecture and no handling of critical non-technical risk. And the underlying belief is that requirements can be understood with minimal technical experimentation and that requirements changes are not the norm, but something to avoid at all cost.

If one is adamant in wanting to continue using a waterfall approach, one fairly obvious approach is to scope the projects more tightly. As it is much easier to succeed doing one or more small projects than a large monolithic one, this will help the chances of the project succeeding. This is difficult to do in a tendering environment, unless the supplier can influence the tendering process in some way or the customers realise the advantages of doing smaller projects. And of course this solution doesn't scale, if e.g. the customer really needs the big, complex solution.

A more comprehensive solution would be to focus on the missing architectural baselining and move this activity into the engineering stage. This baselining activity can either be incorporated as a part of the contract and done by the supplier (sharing the risk) or, if the contractual constraints on change are strict, done by the customer as part of the requirements elicitation activities. It is very risky to do a fixed price project based solely on the requirements specification as written by the customer. If an offer is given without resolving architectural risks, the chances of correctly estimating total cost of the project are not great. The solution is thus to have the engineering stage of the process to incorporate the architectural baselining activity and it should then be possible to do a fixed price project with a bit more assurance than if the project only relies on the detailed, fixed requirements.

This is a temporary measure in that it moves some of the risk resolving into the first part of the project, but does not solve the deeper problem of having what amounts to a waterfall process with requirements specifications frozen at signoff instead of using a iterative process. The reason this seems to be a sufficient short term solution in that it focuses on the architectural risks and how to resolve these and these are crucial for the successful completion of a project. This can also be a difficult solution in that the customer might not want to move such a big part of the development process into the time/material based part of the project fearing to loose control with time and expenses. This is a technical solution and is, as such, not enough to solve what is fundamentally is a cultural issue.

Any short term solutions should focus on minimising the overall risk of the project without antagonising the customer or the supplier. It is of vital importance in such a situation to be aware of how much it is possible to change in the way projects are driven and contracts are specified. It is better to find and implement the two most important best practices than fail implementing them all. We feel that iterative development and risk resolving (especially architectural risk resolving) are very important and will endeavour to do these even in a fixed requirements environment if at all possible, so that when problems arise they will do so early in the project.

Long term: Change of culture (towards an iterative process)

In the long term, it is necessary, in order to gain any substantial profits from using a process like RUP, to integrate the risk-driven, iterative process model in the organisations. Contracts will have to reflect that it is very risky to try to initialise a fixed price project before the construction phase, where requirements and architecture have stabilised. This is not easy, but maybe the continuing software crisis will help improve awareness of the importance of using processes that incorporates requirements management and risk aversion.

And the software crisis is still here: The Danish Audit Department recently made a survey of 124 public projects, started or finished in the period of 1997-1999, where the total cost pr. project should be min. 2 mill. DKr. All offers were given using the K33 standard contract. The total cost for all 124 project was 4.527 mill. DKr. or on average 37 mill. pr. project. 18 of these succeeded (meaning they where on time and budget). 88 projects either ran over estimate or will not be finished at all. The remaining 18 are not finished yet but seem to be successful.

There is a changing awareness of the inadequateness of the existing contracts and the processes supporting these. [11]

This changing awareness will hopefully result in a move towards a process that is more realistic in terms of not trying to fix price and cost before requirements and technologies are explored in sufficient detail.

If such a change is to happen the software development culture of both supplier and customer will have to change. It is necessary for the customer to accept that it is not possible to fix requirements once and for all. In any project of a certain size an iterative approach will have to be used to minimise risk. On the other hand, if this is to happen the customer will want to be able to stop a project, that is getting out of hand, very fast. It is thus necessary to provide the customer with frequent feedback, i.e. not only an iterative but also an incremental process. To avoid iterating forever the customer needs measurable checkpoints at predefined mileposts. The customer gains more control of the process and is able to react to changes in the business domain in a controlled way. The supplier gets to work in more realistic terms and to build up trust between themselves and the customer. Of course if new forms of process are seen as necessary then changes to the contractual templates will have to follow. These will have to reflect the inherently iterative nature of the processes and incorporate the feedback activities in the contract.

Conclusion

The contract situation nowadays involve meeting very rapidly changing requirements, developing complex solutions and generally balancing the software solution with the customer-business to make the software not only support, but also develop the core business of the customer organisation.

To meet these challenges modern software development processes have become oriented towards iterative software development and risk-management that makes it possible to meet the dynamic requirements of the customers. This is a major step towards maturing software development and making it a cost-efficient and quality-oriented.

RUP is one of the iterative software processes that we have experience with. It’s based on best practices drawn from a lot of successful organisations and is a very flexible process. But in order to broaden the scope of the process, some adaptation of the process is necessary. The public procurement process is not based on the idea that software development is dynamic and change is everywhere, but rather the opposite.

To return to the glass slipper: It is not the slipper that is too small but the foot that is too big. But only resizing the foot is too drastic a measure.

Our example of how to handle public procurement is obviously one area that we find is very important to make processes work in accordance with the customer's needs. It is based on our own and our customers experiences working with a modern iterative software development process while also meeting the requirements of fixed-price contracts

The transition to modern software processes and technologies necessitates several culture shifts that will not be easy to achieve. Some of these change will be resisted by customers or by factions within a project or organization. Therefore we think its necessary to adapt the process to the customers world and changing the development culture incrementally. Our recommendation is: Focus the first shift in culture on the architectural baselining.

Our experience tells us that, if the customer agrees to let the supplier do architectural baselining on a time/material basis instead of a fixed price contract, the rest of the iterative approach is more likely to become a success. Otherwise the customer has to pay for not sharing the risk with the supplier in terms of very conservative estimates, resulting in a very expensive offer for the project or in costly overruns or low quality.

References

[1] The brothers Grimm: *Cinderella or the little Glass Slipper*

[2] RUP online - Rational Unified Process version 2000.02.10

[3] Philippe Kruchten: *The Rational Unified Process: An introduction 2nd ed.*, March 2000

[4] The OPEN methodology Henderson-Sellers, B., 1996, Object Magazine (Nov 1996), 6(9), 56-59 (+ figures)

[5] Kent Beck: Extreme Programming Explained - Embrace Change, 1999, Addison-Wesley Pub Co.

[6] DSDM: Dynamic Systems Development Methods

[7] Ivar Jacobson, Grady Booch, James Rumbaugh: *The Unified Software Development Process*, January 1999

[8] PLS Rambøll: *IT-Købekontrakt 2nd ed.*, 2000

[9] TietoEnator Consulting: *EU settings*, April 2000

[10] Watt S. Humphrey: *Managing the Software Process*, 1989

[11] Danish Data Society: *Contracts regarding complex IT - mostly for outsourcing*, 2000

Company: TietoEnator

With a staff of 10.000 in 13 European countries and an annual turnover of 1.2 billion euros TietoEnator is a leading European provider of high-value-added IT services.

The Group specializes in developing and managing its customer's business operations in the emerging Network and Information Society. Most of the products and services in this new world are produced, distributed and consumed digitally via networks. TietoEnator is playing an active role in building this global society.

TietoEnator aims to be a strategic IT partner to its customers. This requires focusing on businesses in which the company can achieve superior expertise and, in this way, offer significant added value to its customers.

TietoEnator's services are consulting, development and integration of IT systems, operation and network management and software products. The scope of services covers everything from the supply of single information systems to taking responsibility for a customer's entire IT operation.

Author: Annette Ibsen, Senior Consultant in TietoEnator Consulting A/S.
Education: Software engineer from Odense Tekniske Skole, Denmark, July 1984

I have been working with object technology methods since 1990 in different kinds of business areas, mainly as a method specialist. Now I'm working with Modern Software Processes and Software Process Improvement.

In 1999 I got certified as instructor in Rational Unified Process by Rational Corporation and have since worked with implementation of Rational Unified Process.

Author: Jesper Rugård Jensen, Consultant in TietoEnator Consulting A/S.
Education: Cand. Scient. Soc. from Roskilde University, Denmark 1992. Certified Sun Java Programmer. Certified Rational OOAD instructor.

I have been working with object technologies since 1987.

I have been working with TietoEnator since 1997 focusing mainly on the practical sides of object technology analysis/design and implementation. In this capacity I have worked with a range of different customers of all sizes implementing and teaching object technology including RUP and OOAD.

Before this I have taught on commercial colleges, including courses on obejct oriented methods at Copenhagen Business College.

Session 13 - SPI in Small Businesses

Session Chair:

Yingxu Wang,

IVF

Support for Effort Estimation in Small Software Companies using Bayesian inference

John Moses,
University of Sunderland, School of Computing, Engineering and Technology,
P.O. Box 299, Sunderland, SR60DD, United Kingdom
E-mail j.moses@computer.org

John Clifford,
STCS Ltd., Gear house, Salt Meadows Rd, Gateshead, United Kingdom.

Abstract

A description of a small software company's effort estimation procedures and their requirements for an effort estimation improvement process are discussed. Support for small software company's effort estimation requirements is then proposed. The support takes the form of Bayesian statistical models and a data collection procedure that will allow the development of the models. The models will be used by effort estimators to identify factors that interact with effort estimates and to learn how to improve estimation consistency. Further, a model can be developed that can take into account the need for accuracy in both original estimates and final estimates that include a contingency allowance. The statistical models can also be used to help improve estimation in familiar and unfamiliar problem and solution domains

Keywords *Estimation effort, Bayesian inference, subjective miss-classification, ordinal scale estimation consistency, small companies, interactions with factors, contingency allowance, estimation accuracy.*

1.0 Introduction

The use of personal memory and guessing by estimators remain popular aspects of approaches to effort estimation, [12]. Estimates given in this way are subjective. Even effort estimates provided using any of the current effort estimation techniques are inherently subjective, [4,7]. However, improvements in estimator's estimation accuracy are needed, [9]. We can describe estimators' estimates by using Bayesian inference approaches [2,13,16,18]. Bayesian inference can give rise to calculations for mathematical integrals that at one time were considered intractable. However, computer simulation techniques, for example the Gibbs Sampler, are now available to perform such computations, [6,17]. A Bayesian approach can enable the production of individual estimator error distributions over an ordinal effort scale of measurement. The error distributions can then be used to allow estimators to learn how their estimates differ from either a consensus estimate from a group of estimators or from the actual system development effort. This learning process could be used to improve an estimator's accuracy.

Further, with Bayesian inference it is also possible to determine whether an estimator is unduly influenced in their estimation by factors inherent in the estimation process. For example, given appropriate data it is possible to establish whether there is evidence that the factor ‘program language’ influences the chance of obtaining a correct estimate for a particular estimator when it does not affect another estimator’s estimate. The ethos of this approach is comparable to that of determining if the size of a module is a predictor of estimate error for the module as exemplified by the more usual Frequentist statistical approach adopted in [7]. However, the Bayesian approach enables us to see how each individual rater is affected by factors that might be considered to influence effort estimation. The Bayesian approach also provides error distributions that can be used as a tool to aid future estimation for each estimator under the appropriate set of factors, which the Frequentist approach in [7] will not. This Frequentist approach tells us if a factor is a likely predictor of an over estimate by 35% or under estimate by 35% or whether it is within 35% of the actual value. Factors include module size, which must be estimated when tendering or at requirements specification. This approach can be used to predict the difference between estimate and actual values on a three classification error scale, and thus if the factor may be considered as a predictor of over or under estimation. However, we cannot tell if the influence varies over different levels of effort estimate. Further if a factor (e.g. module length) is estimated inaccurately then we may easily get an inaccurate estimate error prediction.

A Bayesian approach will tell us how miss-classifications vary over the effort estimate scale. It will tell estimators how likely they are to have miss-estimated for the particular effort value they have arrived at. The estimator can then review their estimation procedure and using their memory and analogy they can make adjustments if necessary to improve the estimate.

2.0 Pragmatism in effort estimation and data collection in small software companies

We propose that the characteristics of small software development companies require an approach to supporting effort estimation compatible with their working practice. Further each company’s estimation procedure may be appreciably different. Small companies current approaches may not be optimum, however they are likely to perform better than most if not all current prediction systems, [9, 12]. Therefore, any method of support must allow small companies to use their own estimating approach. Hence, the method of support must be flexible enough to allow tailoring to an individual company.

To remain competitive small companies do have to go through an effort estimation procedure even though it may be quite informal. They may tend to rely on simple approaches for estimation. Individuals within the organisation are likely to have their own approach to estimation and to use their own adjustment 'factors', [7,9,12]. Further the accuracy of an estimate compared to the actual effort is usually never examined closely, and the opportunity to improve an estimate approach is thereby missed. For example, estimates may be made based on system functionality (and the likely number of software modules) and by making subjective allowances for contingency and integration. Thus without knowledge of an estimator’s subjective tendencies determining were discrepancies have occurred in the estimation process will also be time consuming and difficult.

We noted that all effort estimation techniques incorporate subjective measures. Therefore, we believe our approach to improving estimation can be applied with some modifications to all software development companies regardless of size. However, small companies usually do not have project managers who would devote a significant proportion of their working time to estimation. Small companies may also not possess expertise in as many problem and solution domains as a larger organisation. Although important to large organisations, support in unknown domains will be of major importance to small companies, since they will be confronted more frequently with projects in which they have no expertise. Further, in larger organisations project managers may use more formal and more time consuming estimation procedures, e.g. COCOMO [1]. Such procedures are usually too resource demanding for the estimators to use in a small company. The support we describe in the following Sections takes small companies lack of resources into account.

2.1 The problem and solution domains

Small software development companies use estimates that include the subjective tendencies and past experiences of their employees. In addition, they may be faced with entirely new problem and solution domains. The range of potential projects and their associated factors that a small company may tender for is often large in relation to the size of their work force. For new domains estimates may be made by considering a subjective estimate of how long it would take to develop a system for the problem domain in a familiar solution domain, cf. [7,15]. Some form of contingency will then be added to this estimate. For example, one of our collaborating companies has successfully tendered for a project to set up a web site for a client using Java. The company has no direct experience in this problem and solution domain. However, they have a vast amount of experience in database applications using a variety of DBMS. Their tender was based on the effort required to produce the system in ORACLE RDBMS with contingencies added to that effort estimate. It is also important that estimates should be 'accurate' to within the range of the original estimate plus any contingency allowance that a company might wish to add.

Support for effort estimation in small software development companies is therefore needed in two areas. Firstly, support for effort estimation is needed to improve consistency of estimates when problem and solution domains are known. Secondly, for unknown problem or solution domains some knowledge of how the estimators subjectively adjust estimates will be of value. In the latter case knowledge of estimator subjective tendencies and their classification of the different problem and solution domain factor values may help to improve future estimation both in the new domains and in future unknown domains. That is, knowledge of how they perceive the new problem and solution domain factors and how they tend to estimate to allow for new domains will be of value in future unknown domains. Data collection and statistical analysis over time could reduce the number of unknown domains for a company. Then assistance for estimators will be more closely allied to improving their estimation consistency rather than helping them cope with new problem and solution domains.

2.2 Ordinal scale estimation

An estimator's estimate will be subjective. However, if the estimate is a rational subjective measurement [14] – it should demonstrate consistency. For measurement to take place most of the time the estimators should be able to agree on some value, on an associated scale of measurement (and the error rates should be uniformly distributed) [11]. Since the estimate is subjective an ordinal scale of measurement might be appropriate. Consistency should then be demonstrable on this ordinal scale. At times when some of the estimators' values fall outside of the class, chosen by most of the estimators, we should assume miss-allocation or disagreement between estimators. Choosing appropriate boundaries for the ordinal scale ranges might be based on ensuring that the scale gave consistent measurement and the scale was useful to a particular company's notion of 'accuracy' and to their range of project sizes. Little or no information may be lost in imposing an ordinal scale because estimators' estimates are not generally close enough to the true values to be considered as error on a ratio scale of measurement, cf. [10]. (Compare this approach with the ordinal scales used for development decision support using Bayesian Belief Networks in [3].) However, moving to an ordinal scale could mean a loss of ‘accuracy’ due to miss-classification of final estimates (which include contingency allowances). This problem is considered in Section 6.0.

3.0 Bayesian inference

Bayesian inference uses Bayes Theorem. The theorem states that the probability that two events A and B occur jointly is equal to the product of the conditional probability that event A occurs given B occurs and the probability that the event B occurs, i.e. $Pr(A,B) = Pr(A/B) \times Pr(B)$ [6]. In Bayesian inference there is no fundamental distinction between observed data and the parameters of a statistical model. Bayesians consider them all to be random quantities. Let y be the observed data and θ the model parameters and missing data. Then, from Bayes Theorem the joint probability distribution over all random quantities is $p(y,\theta)$. This joint distribution comprises a prior distribution $p(\theta)$ and a likelihood (sampling distribution) $p(y/\theta)$. The prior distribution embodies our prior knowledge about the model parameters and missing data. The likelihood is regarded as a function of θ for fixed data y. A full Bayesian probability model is given by specifying $p(\theta)$ and $p(y/\theta)$ such that $p(y,\theta) = p(y/\theta) \text{ x } p(\theta)$. After observing the data Bayes Theorem is used to find the distribution of θ conditional on y, $p(\theta/y) \propto p(y/\theta) \text{ x } p(\theta)$. This distribution is known as the posterior distribution of θ.

The primary task of the Gibbs Sampler is to develop a model for the joint distribution $p(\theta,y)$ and perform computations to summarise $p(\theta/y)$ [5].

Essentially, the Gibbs Sampler implemented in the software BUGS [16] can be used to determine the model parameter probability distributions that represent the estimators’ error distributions (the parameters) when the actual effort and the estimates (the data) are supplied to an appropriate statistical model. An appropriate model to use is the 2-Dimensional Multinomial model, [2,16].

The probability distributions (error distributions) used in a multinomial model represent all the possible ways an estimator can miss-classify over a range of effort values. For example, if a twelve class ordinal scale was used to represent effort classification then there are eleven possible incorrect ways an estimator can miss-classify (given the actual effort) and one correct way. Hence, the probability distributions represent the probability of classifying into class one given class one is the actual class, class two given class one is the actual class, etc for the twelve classes. Further we also require the probabilities that the true class is class two, class three etc.. Hence, the multinomial model uses a twelve by twelve matrix of error distributions. Then, for example, error[1,2] would represent the probability distribution that class 2 is chosen given class 1 is the actual (true) class. These distributions summarise the classifications made by an estimator for the given data. With these distributions we then know how an estimator tends to miss-classify. Therefore, given this information an estimator can examine their estimation procedure to see why they might tend to classify in a particular way for a given effort class. See Figure 1.

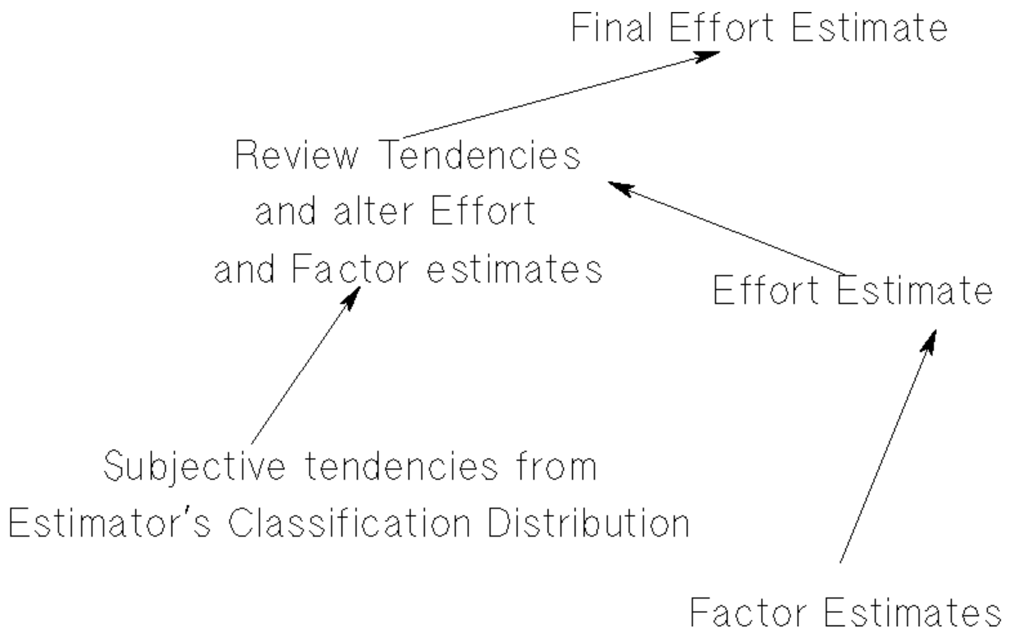

Final Effort Estimate
Review Tendencies and alter Effort and Factor estimates
Effort Estimate
Subjective tendencies from Estimator's Classification Distribution
Factor Estimates

Figure 1 Proposed effort estimation support in a familiar domain

distribution	mean	standard deviation	2.5%	97.5%	median
error[1,1]	0.37	0.12	0.13	0.6	0.3
error[1,2]	0.1	.07	.02	0.27	0.08
error[1,3]	0.05	0.05	0.0	0.17	0.08

Table 1 Error distributions for classification of an effort estimate into class 1, 2 and 3 when the actual effort belongs to class 1.

Table 1 shows some examples of error rate distributions. error[1,1] is the probability that a system estimated as class 1 effort has an actual effort value in class 1. error[1,2] the probability of miss-estimating an actual class 1 effort system as class 2; and error[1,3] the probability of miss-estimating an actual class 1 effort system as class 3. (The example has been developed using data taken from The COCOMO Software Project Data Base [1, page 496].) For each distribution the mean, median, standard deviation and the 95% confidence limits are given. We can see that for this data the mean value of miss-classification of a class 1 effort system into class 2 is 0.1; further the mean probability of correctly estimating a class 1 effort system is about 0.4.

In addition, we can develop 3-Dimensional multinomial models that give us distributions for miss-classification given the true class and a factor. Factors of interest might include program language type, problem type, solution domain type or contingency allowances.

4.0 Alternative approaches for effort estimation consensus

The Bayesian approach to consensus agreement differs from that of the Weighted Kappa statistic [18]. Kappa does not describe the way in which the estimators miss-classify during measurement. It also does not describe the error distributions for the estimators. Thus it can not help us to assess the estimators' subjective miss-classification tendencies. It is also not very informative about estimators’ performance with respect to 'true' or consensus classifications. Hence, it cannot tell us how estimators are likely to miss-classify during future measurements.

In addition, the Delphi technique [8] or the Wide band Delphi technique [1], are sometimes used to produce a single estimate from a group of estimators. The Delphi technique involves providing estimates that are reviewed iteratively with respect to the group median estimate. The techniques prevent direct inter-group interactions. They therefore avoid influences due to assertive members and political or authoritarian pressures. However, convergence to a single estimate is not assured. Consequently, a range of estimate values may actually be represented by the final median estimate. The estimators learn about their over or under-estimation tendencies from a median estimate. This means that no single estimator may have actually chosen the value. Each estimator may influence the others’ refinements of their original estimates. The influence takes little account of which raters are more consistent estimators. However, it is quite likely that some estimators are more consistent with their estimates in terms of the true value than the others, [18]. It follows that the more consistent estimators may adapt their strategy to meet the median value generated from less consistent estimators. The estimates can be weighted by self-assigned competence scores, [8]. These weights will be subjective, unless past data are examined and a statistical procedure used, for example, one such as described in this study. Using Delphi the estimators may (if learning has taken place) be learning to estimate inaccurately. With Delphi techniques we cannot be sure that estimators can actually agree independently of the technique. Delphi therefore forces convergence on an estimation scale for which consistent measurement (on an ordinal scale) may not be demonstrable.

5.0 Data collection

In order to provide assistance for the smaller company we are using Bayesian inference. In addition our participating Software Development Companies will collect data that are appropriate to their estimation procedures. Data are being collected over several months and in some cases years and continuously analysed during this period. In addition, a further requirement for one of our companies is that the estimators must not spend more than 20 minutes each week logging data for analysis. One of the difficulties for smaller companies in software development is scarcity of resource to collect data. In addition, the introduction of new personnel to perform data collection is prohibited because of competitive tendering and security issues. Research students unfamiliar with the company's working practices would not necessarily be welcomed when the company is under pressure to tender for a project. Therefore, the number of factors in the data set must be restricted to a set that the company's estimators have time to collect. Further, the data set necessary for estimation must be sufficiently focused to provide the required level of accuracy. It must also be small enough so that the estimators would actually have time to arrive at the necessary factor estimates during their limited time for effort estimation. The time needed for an estimator to arrive at an effort estimate may be as short as one day.

One of our collaborating company's current estimation practice involves one and often two separate effort estimates. Estimation takes place at:

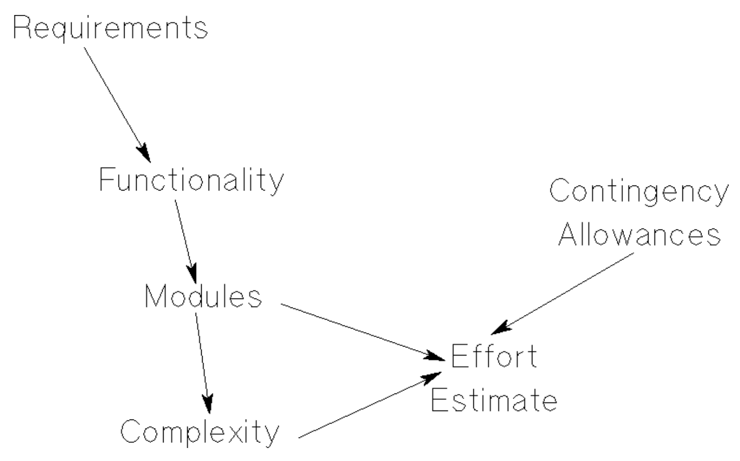
1) Feasibility, this estimate is nearly always produced for every new project.

2) Requirements, when the requirements are more completely understood and in a more formalised state a second estimate is often produced.

The estimates are based on knowledge of the following factors and attributes.

1.The Problem Domain
a) Business sector.
b) System type.
2.The Solution Domain

a) The hardware.
b) The software (program languages)
c) Tools being used.
d) Development methods requested.
3. Customer/User level of knowledge of the Application/Problem Domain, measured on an ordinal scale.
4. Team/Estimator knowledge of
a) The Application/Problem domain.
b) The solution domain: Language and Hardware and other Technical constraints, e.g. execution speed.
5.The estimator ID.
6. The Number of Modules that are likely to be needed.
7.The Module/Function COMPLEXITY for each module - rated on an ordinal scale (simple, moderate, complex, very complex, extraordinarily complex: 1,2,3,4,5)
8.Additional data may sometimes be collected for:-
a) Design Effort
b) Implementation Effort
c) Test Effort
d) Overheads



1.1.1.1.1 Figure 2 Present approach to effort estimation in a collaborating company

Using the factors and attributes 1 to 8 enables an estimator to provide their own effort/time estimate. In order to produce an effort estimate the estimators in the company only indirectly address the concept of project 'size'. For example, one estimator uses the estimated values: number of modules and their complexity (which are derived from the system functionality). See Figure 2. Effort is then estimated by summing the likely development effort (which is estimated using the functionality and complexity of the module given by the estimator) over all the modules:

$$\text{Estimated Effort } = \sum_{i=1}^{M} E_{CM_i}$$

Where, M is the number of modules identified (6), CM_i is module i's complexity (7) and E_{CM_i} is the estimator's estimate of the likely development effort for module i.

The E_{CM_i} are estimated using knowledge of CM_i (7), Tools (2c), Methods (2d) and program language (2b). Hence, system size is not explicitly estimated. Further, the E_{CM_i} will incorporate the estimator's subjective tendencies. Finally, other contingencies are added to the estimate for the other factors, e.g. User knowledge (3), Hardware (2a), problem domain (1), Integration Testing effort (8c) and Overheads (8d) etc.. The final estimate is therefore likely to include other estimator subjective tendencies.

Frequently more than one estimator will provide an estimate for a project. Each estimator interprets the factors using past experience and personal viewpoints. In addition, the estimators will use their own informal contingency allowances for unfamiliar factors, e.g. for estimation in new solution domains, etc..

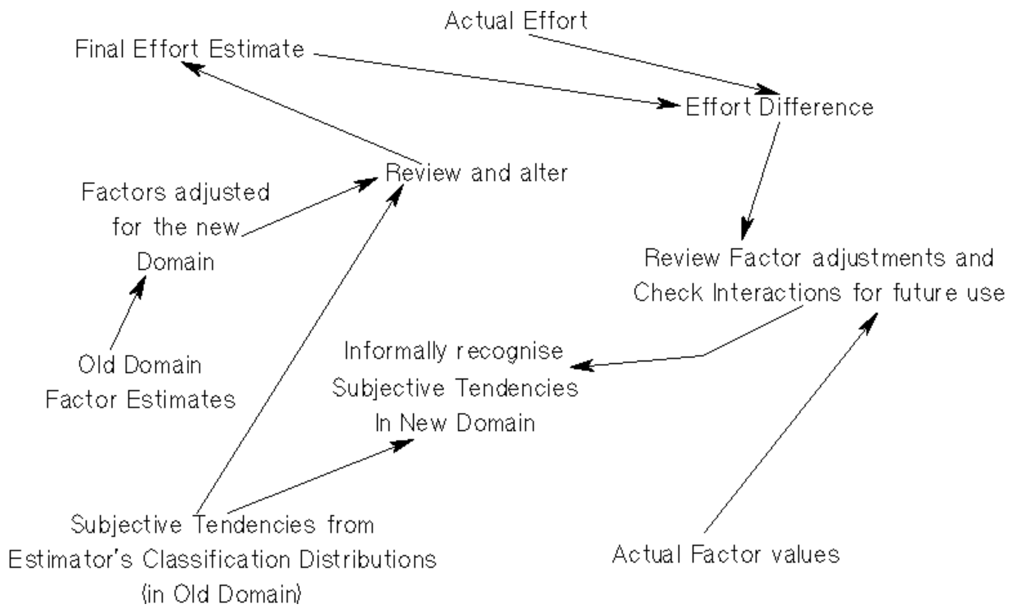

Actual Effort
Final Effort Estimate
Effort Difference
Review and alter
Factors adjusted for the new Domain
Review Factor adjustments and Check Interactions for future use
Old Domain Factor Estimates
Informally recognise Subjective Tendencies In New Domain
Subjective Tendencies from Estimator's Classification Distributions (in Old Domain)
Actual Factor values

Figure 3 Proposed effort estimation support in a new domain

6.0 Improving estimators’ measurement consistency

Using Bayesian inference we can test for independence from factors that are supposed (or not supposed) to influence (interact with) an estimator's estimates. Such factors might include problem or solution domain factors. The raters’ probability distributions for correctly and incorrectly classifying estimates for a given factor can be compared. We can then decide if the estimator's chance of correct classification might vary with a problem domain or solution domain factor, for example.

Knowledge of interactions in old domains may help to decide on the values of similar factors in new domains. For example, in the new domain of web-site development using Java estimates might be based on development in ORACLE. Then, if we had already shown that for ORACLE solution domains the factor screen complexity (module complexity) tended to interact with estimate error rates as follows: for systems of actual effort classes in which simple and moderately complex screens tend to dominate the median chance of correct classification of effort for a given effort class is about 0.7 (distributions of the type error[3,3]). And, if we knew that for systems in which complex and very complex screens tend to dominate the correct classification was low say 0.4 mean with a tendency to under-estimate, i.e. error rates of the type error[3,2] and error[3,1] are quite large for a system of actual class size 3. Then we might begin to estimate for the effort required for the Java site based on these error rates for ORACLE RBDMS development, and make appropriate adjustments in the new domain. See Figure 3.

It is often important to a company that their final estimate with the contingency added on is accurate. Table 2 shows a fictitious example of classes for original estimates and contingencies that might be added to them to form the final estimate. A simple ordinal scale could be miss-leading in these circumstances. For example, if an original estimate were evaluated as 9 weeks it would fall into class 2. Adding on the contingency of 2 weeks would take the final estimate into class 3 (9 + 2). The company might consider their estimate as good if the actual effort was within a range of 'less than 9 weeks to 11 weeks'. Then the chance of correct classification into this range would indicate how accurate an estimator was for this project effort range. In stead, with a simple ordinal scale and a 2-Dimensional multinomial model we would get the chance of miss-classification into class 2 (6 - 10) when the true class is class 3 and the chance of correct classification into class 3 (11 -14) etc.. Hence, if the actual class value turned out to be class 2 (8 weeks say) this would give rise to a miss-classification and we would not be certain if the estimates of this type were inaccurate because the original estimate was precise and the contingency was too large or unnecessary. However, by including a third dimension in the multinomial model for contingency we can (for example) estimate the chance of miss-classification into class 3 when the true class is class 2 and the contingency for class 2 was added to a class 2 estimate to give a class 3 estimate. Hence, for any estimated class the chance that a contingency estimate has taken the original estimate into the next estimated class can be determined. From this we can decide the chance of an original estimate being accurate; and, the chance of the contingency causing an apparent inaccuracy due to the scaling process.

Scale	Original estimate	Contingency
class 1	1 - 5 weeks	1 weeks
class 2	6 - 10 weeks	2 weeks
class 3	11 - 14 weeks	3 weeks
class 4	15 - 19 weeks	4 weeks

Table 2 Fictitious example of original estimates with contingency

7.0 Conclusions and Future Applications

Factors that are estimated by estimators in one of our collaborating companies in order to produce an effort estimate have been illustrated. The factors used may vary from one company to another. Further, all small companies are different in the sense that their estimators are all different people, their project experiences will be different and their perception of projects that they have in common may also be different. Therefore any support approach for estimation must be general enough to cope with this lack of commonality. It must also be flexible enough to enable the estimators to make their own adjustments to an estimate.

We propose that the estimators use the Bayesian inference procedure within both familiar and unfamiliar problem and solution domains to assess the likely accuracy of their estimates. Using the classification distributions from the Bayesian inference procedure an estimator can check to see if they have produced an effort estimate that is likely to be a correct. They can also easily see if the effort class is one in which they are likely to over or under estimate. With this knowledge the estimator may then wish to review the estimates they made for the factors and contingencies. Classification distributions can also be developed for these estimated factors. These distributions can then be used to determine the chance that the factor estimates are correct. The estimators can then assess whether they are confident about the effort estimate and the factor estimates, and review them as necessary, see Figure 1. This procedure is compatible with, but more informative, more convenient and quicker to use than, review approaches used by some estimators in which they examined the characteristics of the new project and other projects with similar effort estimates „ ….. to see if they appeared to have about the same content“ [9].

Data from each newly completed project could be added to the inference procedure to help model any changes in estimator subjective tendencies. Although the data may no longer be thought of as independent random values. (Since the original model that the estimators learned from will be influencing the new project estimate data and therefore the new data should not be considered independent of the model.) However, the changes in an estimator's subjective tendencies could be modelled using Bayesian models that utilise Time Series adjustment [5].

In addition, to the procedure in 6.0 for using interactions from old domains, in unknown problem and solution domains existing error distributions can show how an estimator adjusts their usual subjective contingencies to account for the new domains. They can help to elicit information about the way estimators adjust their contingency estimates when confronted with an unknown solution or problem domain. Estimators can isolate this information using their old domain error distributions and by referring back to the factor and contingency estimates they made for the new domains, when the actual values become available. However, this procedure would be informal and requires that the estimators record why they made adjustments to factors and contingencies in each domain at the time when it was considered new, see Figure 3. Further, using other 3-Dimensional models it is possible to assess how accurate estimators are with respect to their contingency allowances, and to investigate when these contingencies need adjustment.

Acknowledgements

The author acknowledges the support provided by the ESRC (UK) and STCS Ltd., Gateshead, UK for access to publish their effort estimation factors. The authors are also grateful for the referee’s helpful comments.

References

[1] Boehm, B.W., Software Engineering Economics, Prentice-Hall, New Jersey, 1981.

[2] Dawid, A.P., Skene, A.M., Maximum Likelihood Estimation of Observer Error-rates using the E-M Algorithm, Applied Statistics, 28, No. 1, pp. 20-28, 1979.

[3] Fenton, N., Neil, M., A Critique of Software Defect Prediction Models, IEEE Trans on S.E., Vol. 25, No 1, September/ October, 1999.

[4] Fenton, N.E., Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach, 2nd Ed., PWS Publishing, 1997.

[5] Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B., Bayesian Data Analysis, Chapman & Hall, 1998.

[6] Gilks, W.R., Richardson, S., Spiegelhalter, D.J., Markov Monte Carlo in Practice, Chapman & Hall, 1996.

[7] A. Gray, S. MacDonell, M. Shepperd, Factors Systematically Associated with Errors in Subjective Estimates of Software Development Effort: The Stability of Expert Judgement, Sixth IEEE In. Symposium on Software Metrics, 4-6 November, 1999, Boca Raton, Florida.

[8] Olaf Helmer, Social Technology, Basic Books Inc., New York, 1966.

[9] Hughes, R.T., Expert judgement as an estimating method, Information and Software Technology, Vol. 38 (1996), pp. 67-75.

[10] Kitchenham, B., Pfleeger, S.L., Fenton, N., Towards a Framework for Software Validation, IEEE Trans. on S.E., vol. 21 no. 12, pp. 929-944, December 1995.

[11] Kyburg, H.E., Theory and Measurement, Cambridge University Press, 1984.

[12] Lederer, A.L., Prasad, J., A Causal Model for Software Cost Estimating Error, IEEE Trans. on S.E., Vol. 24, No. 2, February pp. 137-147, 1998.

[13] Moses, J., Bayesian Probability Distributions for Assessing Subjectivity in the Measurement of Subjective Software Attributes, Information and Software Technology, Vol. 42, Issue 8, 15 May 2000, pp. 533-546.

[14] Roberts, F.S., Measurement Theory with Applications to Decision Making, Utility and Social Sciences, Addison-Wesley, 1979.

[15] Shepperd, M., Schofield, C., Estimating Software Project Effort Using Analogies, IEEE Trans. on S.E., Vol. 23, No. 11, November 1997.

[16] Spiegelhalter, D.J., Stovin, P.G.I., An Analysis of Biopsies Following Cardiac Transplantation, Statistics in Medicine, Vol. 2, pp. 33-40, Pub. J. Wiley & Sons, 1983.

[17] Spiegelhalter, D.J., Thomas, A., Best. N., Gilks, W., BUGS 0.5, Bayesian inference Using Gibbs Sampling Manual (version ii), MRC Biostatistics Unit, Cambridge, August 1996.

[18] Wilson, M.E., Williams, N.B., Baskett, P.J.F., Bennett, J.A., Skene, A.M., Assessment of fitness for surgical procedures and the variability of anaesthetists' judgements, British Medical Journal, Vol. 1, pp. 509-512, 23 Feb. 1980.

Curricula Vitae

John Moses is a Senior Lecturer in the School of Computing at the University of Sunderland. He has received a BSc. degree in Applied Mathematics and Computing Science from the University of Sheffield, an MSc. in Operational Research from the University of Hull and a PhD. in Computer Science from the University of Sunderland in 1997. He has worked in computing as a systems analyst in the steel, water and plastics industries and as a project manager for data-capture systems. He has lectured at the Universities of Teesside, Humberside and Sunderland. He is a member of the British Computer Society, the Operational Research Society and is an affiliate of the IEEE computer society. His research interests are in software engineering and software measurement.

John Clifford is Managing Director and one of the co-founders of STCS Ltd., Gateshead, Tyne and Wear, UK. He has over thirty years experience in the Computing industry including time spent as a Project Manager and as a University Lecturer. He has developmental experience across a wide range of client companies and for a wide variety of project sizes.

STCS Ltd. is a Software Development Company located in the North East of England. STCS has a large and varied range of client companies of all sizes in the private and public sectors.

EVALUATION OF SMALL SOFTWARE COMPANIES FOR LARGE CONTRACTS

Elif Demirors
The Bilgi Group Software Research, Education and Consultancy
Ankara Turkey
demirors@bg.com.tr,

Onur Demirors
Informatics Institute, Middle East Technical University
Ankara Turkey
demirors@metu.edu.tr

Introduction

The global software industry includes a considerable number of small companies. In Turkey, for example, 69% of software companies have 20 or less employees [1]. Similar observations are published for other European countries. Somewhat contrary to this, in large government contracts it is usually the tradition to call in large, mature software developing organizations in order to ensure the completion of the contract and a high quality product. However in some cases large organizations may not be interested in the domain of contract, or the government agency would like to involve small organizations to increase competition. Although generally away from standards, and improvement frameworks, small companies have a potential to develop quality software, especially in their specific domain of expertise. Also, if such a company has a group of very talented, open-minded developers that are ready to change given the chance, then it will have a chance to succeed in contracts much higher than their revenues.

When an agency is faced to select the best-fit software developer among a number of small-sized companies, it fails to find an applicable evaluation strategy. The widely used assessment methodologies, SPICE and CMM, work well in large organizations. However little data has been gathered on the quality of their feedback when applied to small organizations with practices far away from what CMM or SPICE expect to find.

We have been involved in a project that was initiated by the Ministry of National Education (MONE) to define a set of pre-qualification criteria that will evaluate the capability of software development companies. Specific domain of application was instructional software which eliminated most of the large software development companies operating in defence, and finance markets. The experts in instructional software was, and still are, small businesses which involve both software developers and instructional domain experts (mostly teachers). Although the agency was willing to work with the small companies in contracts approximately 1.2M USD each, they wanted to eliminate the businesses with higher chances of failure.

The project took four weeks, and involved software quality as well as instructional technology experts. A set of criteria was formed to be used by the Agency to establish a short list of potential software contractors. Several issues were addressed for defining the criteria: regulations and early experiences of the Agency; regulations and recommendations of the World Bank (the project was financed by the World Bank); the capability of developing quality instructional software; and related best practices and standards of global software industry. The process followed to define the criteria is in accordance with the ISO/IEC 14598-1:1999 Information Technology-Software Product Evaluation.

Methodology

Several issues are addressed for defining a set of pre-qualification criteria that will evaluate the capability of software development companies:

- The regulations, recommendations and early experiences of the MONE- Since the MONE has been involved in acquisition of instructional software packages for more than a year, and has already completed a bidding cycle for off-the-shelf packages, a large experience base has been formed regarding to the company profiles, acquisition process, and evaluation process. This experience base has been utilized in defining the pre-qualification criteria.
- The regulations and recommendations of the World Bank- The procurement of customized instructional software development is required to follow the World Bank's acquisition policies. The strategies of the World Bank for pre-qualification of bidders and the regulations on the bidding process have been considered. The pre-qualification criteria support the strategies and policies of the World Bank.
- The capability of developing quality instructional software- Instructional software development requires software engineering expertise, subject matter expertise, and instructional design expertise. The capability of a company to operate in this domain depends on how well these different expertise areas are represented, and how well individuals from different backgrounds are harmonized. Suggestions of the industry standards as well as the World Bank policies on the infrastructure required for a successful instructional software development project are taken into account in defining the pre-qualificaiton criteria.
- The related best practices of the global software industry- Software engineering discipline has been working on standardization of related practices independent of the application domain. Both the International Organization for Standardization (ISO) and the Institute of Electrical and Electronics Engineers (IEEE) published a range of standards addressing acquisition and evaluation issues in software development. Additionally, various recommended practices as well as interpretations have been published by these

organizations and by other agencies. The pre-qualificaiton criteria defined are based on the requirements and suggestions of such publications.

The process followed to define the pre-qualification criteria is in accordance with ISO/IEC 14598-1:1999 Information Technology- Software Product Evaluation. First, the purpose of evaluation was clarified, and then, the types of products that need to be evaluated and the quality model that defines the relevant quality characteristics are identified. The process then focused on selecting the metrics for evaluation and establishing the rating levels for these metrics. The criteria for assessment were also defined. The remaining phases of ISO 14598 evaluation process are out of the scope of this study.
As mentioned above several international standards and practices have been utilized in this study. These are:
(a) ISO/IEC 12207:1995 : Standard for Information Technology – Software Life Cycle Processes.
(b) ISO/IEC 14598-1:1999 Information technology - Software product evaluation -- Part 1: General overview.
(c) ISO/IEC DIS 14598-4 Information technology -- Software product evaluation -- Part 4: Process for acquirers.
(d) IEEE Std 1062:1998 : IEEE Recommended Practice for Software Acquisition.
(e) ISO/IEC TR 15504: 1998 Information Technology - Software Process Assessment.
(f) IEEE Std 1012:1998 : IEEE Standard for Software Verification and Validation.

Organization

The measures established for pre-qualification of companies are classified under three equally important criteria: organizational capability criteria, product capability criteria, and process capability criteria. For each one of these groups, a set of quality characteristics are defined. The quality characteristics are composed of a set of attributes that are measured in terms of the indicators listed.

A company's total score is calculated by summing up the scores that the company earns from each of the three capability criteria. The score of each criteria is obtained by totaling the scores gained from each attribute as defined by the attribute's measure in terms of the indicator values. In other words, the evaluation will determine whether a specific indicator is valid for that company and if so, the corresponding point value will be added to the company's score according to the measurement policy defined. A company is also required to earn a score above the threshold defined for each criteria for pre-qualification.

The hierarchical organization of the pre-qualification criteria is depicted in Figure 1.

Capability Criteria

Organizational Capability Criteria

The organizational capability criteria characterizes the adequacy of organizational infrastructure in supporting the development team during a customized instructional software development contract. For a software development project to be successful a solid infrastructure should exist that includes experienced personnel, mature business practices, adequate financial resources and appropriate development environment. The organizational criteria is divided into three characteristics as defined below:

- Company Profile
- Staffing
- Development Environment

The weight of the organizational criteria is one third of the overall score, that is 300 out of 900. There is a threshold of 150 points over 300 for pre-qualification.

Product Capability Criteria

The product capability criteria evaluate the quality of an existing product to determine a company's past performance in software development which is an important indicator of its future performance. For evaluation, the bidder is expected to submit one of its products that best satisfies the requirements listed. If there is only a prototype of a future product, then the bidder may also submit this prototype for evaluation. The size of a product or prototype is required to be no greater than the size of a typical instructional software targeting a specific course for a specific grade (i.e. Mathematics for the 6th grade).

The evaluation is performed on the product itself and on the intermediate documents produced during the development process, namely software requirements specification, design documents, and test documentation. The success of the product in the market will also be evaluated if the company will be able to submit appropriate documents.

Product capability criteria is divided into three characteristics as defined below:

- Product/Prototype Quality
- Quality of Intermediate Products
- Customer Satisfaction

The weight of the product-in-use evaluation is one third of the overall, having a score of 300. There is a threshold for this part of evaluation, that is 150 over 300, for pre-qualification.

Process Capability Criteria

A company's software process capability determines its ability of producing quality products since product quality depends on the quality of the processes enacted during development. The process capability criteria is generally consistent with the Software Engineering Institute's (SEI) Software Capability Evaluation methodology which evaluates process maturity with respect to the framework defined by the Capability Maturity Model (CMM).

The software process capability is evaluated with a Maturity Questionnaire. The Maturity Questionnaire is based on SEI's maturity questionnair and is being used as the formal data collection device within the Software Capability Evaluation methodology. Instead of requesting a full Level 2 or Level 3 compliance from the bidders, a number of the key process areas are selected based on the requirements of the ministry and on the nature of instructional software development. The key process areas grouped under Basic Management Processes include most of the Level 2 process areas except "Requirements Management" (classified as Extended Process) and " Subcontract Management" (eliminated completely from the evaluation). On the other hand, Extended Processes include the key process areas from Level 3 as well as "Requirements Management". These processes are not considered mandatory like Basic Management Processes but rather taken as indicators of more mature software development process. The ministry reserves the right to perform and/or request on-site independent assessment. If assessment is to be performed during pre-qualification process than all acceptable bidders will be evaluated. However the ministry may choose to perform and/or request an independent assessment within the post-qualification framework in which only prequalified bidders will be evaluated. Any inconsistency with the answers provided here will result the disqualification of the bidder from submitting any bid to this and any related calls.

The process capability criteria is divided into two quality characteristics:

- Conformity to Basic Management Processes

· Conformity to Extended Processes

The weight of the process capability criteria is one third of the overall score, that is 300 out of 900. At least 75 points need to be earned from the key process areas of the "Basic Management Processes" quality characteristic for pre-qualification. There is no threshold for the second characteristic, however any points earned from extended management processes will be totaled to the score of process capability criteria.

Lessons Learned

The project helped the whole team to merge two different worlds of instructional domain and software engineering for the benefit of small organizations which work in between these domains. The criteria defined provides a general guidance on the expectations of the MONE from the suppliers, as a result will also be utilized by the organizations as a guidance for improvement. Our experience and issues that we have learned from the project are summarized below:

- Supplier processes should be the focus but product, people and organizational aspects should also be considered. The organization and product criteria enabled evaluation of the companies previous assets in terms of people, customers and development environment.
- Mature supplier processes best work with mature acquisition processes but acquirers might not be ready for a complete process improvement initiative. As a result suppliers need to adapt their process for immature acquirers. Using objective pre-qualification criteria enables acquirers to minimize risks at early phases without substantial investments.
- International standards should be adopted whenever possible but enforced standards should not prevent organizations to apply different methodologies and techniques. The utilization of international standards enables wide range of organizations to compete. It also enables faster dissemination of knowledge and higher possibility of creating a nationwide acceptance. However many application domains are distinct. For example, instructional software development is generally considered as an art, and engineering of products is often neglected. Work product and process standards should be flexible enough to cover such domains.
- Utilization of suitable pre-qualification criteria is most beneficial if it is combined with an explicitly defined development process and product expectations as well as a well defined feedback mechanism throughout the contract work.

References

[1] TURSOFT, The Turkish Software Industry (in Turkish), 1996.

1. ORGANIZATIONAL CAPABILITY
1.1. Company Profile
ORG-1. Business Experience
ORG-2. Domain Experience
ORG-3. Financial Stability
1.2. Staffing
ORG-4. Existence of Project Management Expertise
ORG-5. Existence of software Development Expertise
ORG-6. Application Domain Expertise
1.3. Development Environment
ORG-7. Adequacy of Software Tools
ORG-8. Adequacy of Hardware Tools

2. PRODUCT CAPABILITY
2.1. ProductQuality
PRD-1. Instructional Quality
PRD-2. Multimedia Quality
PRD-3. User Manual Quality
2.2. Intermediate Product Quality
2.2.1. Quality of SRD
PRD-4. Consistency of SRD
PRD-5. Completeness of SRD PRD-6. Readability of SRD
PRD-7. Testability of SRD
2.2.2. Quality of SDD
PRD-8. Completeness of SDD PRD-9. Readability of SDD
PRD-10. Testability of SDD
2.2.3. Quality of STD
PRD-11. Correctness of STD
PRD-12. Completeness of STD
2.3. Custom. Satisfact
PRD-13. Sales Performance
PRD-14. List of Current Users

3. PROCESS CAPABILITY
3.1. Conformity to Basic Processes
PRO-1. Adequacy of Software Project Planning Process
PRO-2. Adequacy of Software Project Control Process
PRO-3. Adequacy of Software Quality Assurance Process
PRO-4. Adequacy of Software Config. Management Process
3.2. Conformity to Extended Processes
PRO-5. Adequacy of Requirements Management Process
PRO-6. Adequacy of Organization Process Definition Process
PRO-7. Adequacy of Training Program Process
PRO-8. Adequacy of Software Product Engineering Process
PRO-9. Adequacy of Peer Reviews Process

Fig. 1. Pre-Qualification Criteria

A Comparison of Techniques to Support Small Companies with Software Process Improvement

Ebba Thora Hvannberg
Helgi Thorbergsson
Gudrun Agusta Johannsdottir
University of Iceland, Reykjavik, Iceland

Introduction

Small companies with limited previous experience in software process improvement (SPI) often find it difficult to start to improve their processes. There are many hindrances that have to be overcome. One is that with the pressure to deliver products fast to the customer, management finds it difficult to see concrete benefits of process improvement. This leads to lack of commitment on their behalf that again leads to few human resources invested in software process improvement experiments.
For the past two years the ESPICE project (http://www.espice.hi.is) has been supporting Icelandic companies with software process improvement. In this paper we describe and contrast several techniques that we have used. The techniques that we have offered are a mini-PIE club, several half-day workshops, courses, process improvement experiments conducted by a tutor, and web-resources. We asked participants at these events to evaluate the support and will base our comparison on that data. In conclusion we present solutions that are improvements of the techniques described.
We aim to share our experiences with other consultants that are supporting client companies initiating software process improvements. We expect that there are many regions like ours, where formal SPI support has not been particularly strong. The paper can also be guidance for companies that are selecting the type of support that is most suitable when improving their software processes.

Context

Before we assess the needs of our target companies and institutions, we briefly describe the context in which our customers conduct business.

Industry

Icelandic companies and departments developing software are the targets for our services. According to the Statistics Iceland there are 200 companies in Iceland that develop software [1]. Their export in 1998 is estimated about 29 MEURO compared to 0,34 MEURO in 1990 [2]. The export of the local software companies in 1998 estimates to about 1% of the total national export.
Most software companies in Iceland are small, ranging in size from a few developers to several dozen people. Only a few companies in information technology have over 100 employees, but many of them are growing rapidly, merging or operating in a holding. Larger companies often operate their own information technology departments developing software as needed. In 1998 around 4000 people were working in information technology in Iceland, which is roughly twice as much as in 1993. Thereof, 44% work in software development and consulting, 30% in telecommunications, 23% in retail, and 3% in IT industry. [2] [3] [4]
Icelandic software houses are increasingly exploring the possibilities of exporting software products and expertise. This is mainly due to the small size of the domestic market. Many seek the possibility for co-operation with foreign companies, through partnership or joint ventures, in order to secure capital and wider distribution of their products. Foreign companies operate freely in Iceland either as subsidiaries or through local service agents. These include for example Microsoft, Oracle, Software AG, Informix, CA, Compaq, Digital Corporation and Novell and many more. [3] [4]

Government

The Icelandic government publishes a handbook of acquisition [5] for buyers of information technology products, both hardware and software. There are no requirements on ISO 9000 certification but it is pointed out in the handbook that more and more companies manage quality. In the chapter on software, developers are encouraged to review design and source code. Furthermore, the sections on documentation management and configuration management refer to ISO 9001 and to ISO 9000-3.

Education

In 1998, 32,1% of people working in software development and consulting had university degrees [2]. The University of Iceland offers B.Sc. and M.Sc. degrees in Computer Science. Since 1999, The University of Iceland has offered a 45-credit diploma program in Systems Operations. In 1998 over 400 people had graduated from the university with B.Sc. degrees in Computer Science. The Institute of Continuing Education of the University of Iceland offers various courses on Information Technology, including quality management. Reykjavik University has offered associate degrees in Data Processing for a number of years and graduated its first B.Sc. students in Computer Science in 2000.

Needs and maturity of the industry

A survey of the local software industry was conducted by the ESPICE project in March of 1999. A questionnaire was sent to 31 companies that are engaged in software acquisition or development. Twenty one companies responded to the survey, which amounts to approximately 10% of the Icelandic software industry.
The results of the ESPICE survey indicate that many software companies believe they are currently doing software process improvement. Most of them have created in-house procedures, but few if any use international standards to tailor their actions. The standards are by many believed to be too rigid or inflexible to be successfully applied directly to the Icelandic software industry of small to medium enterprises (SMEs). It is however encouraging to see that most of the companies participating in the survey were aware of the necessity of well defined and managed software processes. This was particularly apparent in companies that were growing fast in terms of both number of employees and variety of projects. The fact that the market does not demand quality certification from its suppliers,

inapplicable standards, and lack of time and resources are the most common reasons given for not pursuing SPI in a more structured or determined fashion.
The survey showed that there is a demand for seminars or workshops on SPI related material like best practices, use of standards, SPI awareness, and the development process in general. Some expressed interest in case studies and thought it useful to hear about successful deployment of SPI. Companies, playing the role of the acquirer, expressed interest in seminars and consulting regarding the acquisition process, indicating that these companies are more aware of the demands and the constraints they can express towards their suppliers. Fewer see the need for direct consulting especially in relation to quality certification like ISO 9000.

Techniques for Supporting Software Process Improvement

In order to meet the needs of our clients we have offered several types of services to support the companies with software process improvement. The following subsections describe these techniques with most emphasis on the mini-PIE club and the individual workshops. Other techniques such as courses, consulting and web are also described. Each section evaluates a single technique's usefulness to the participants and suggests ways in which it can be improved. The members of the SPI club were interviewed but the evaluation of the individual workshops is based upon a questionnaire that was sent out to participants. The ultimate goal is to find out what technique will result in most improvement of a software process at a minimum cost.

Mini-PIE Club

At the onset, our objective was to reach out to a number of companies and support them in a step by step manner to perform process improvement experiments. This was the first type of service we advertised. After an invitation was sent out to ninety companies and departments developing software, nine companies indicated interest in joining the club. Five companies became active participants. The mini-PIE club was organised around modules that were held approximately every other month. Figure 1 shows how the modules are meant to be practical and generic but tailored in part to the needs of the participants.

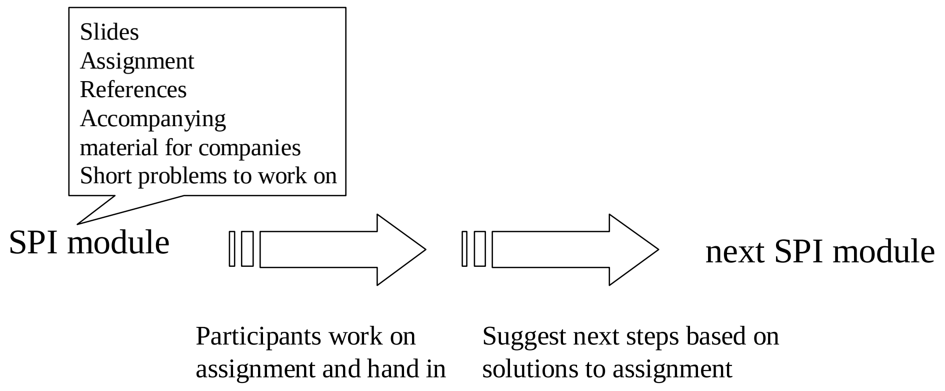

Slides
Assignment
References
Accompanying material for companies
Short problems to work on
SPI module
next SPI module
Participants work on assignment and hand in
Suggest next steps based on solutions to assignment

Figure 1 Functionality of the SPI modules given to the mini PIE club

In between modules, participants worked on assignments that they could send to organisers. Each module was focused around one theme. A list of the modules appears in Table 1.

Module
Kick-off meeting of PIE-club
Introduction to Software Process Improvement
Software Best Practices

Informal meeting of PIE-club, progress assessment
Measurements and Metrics
Dissemination of PIEs

Table 1 Mini-PIE club modules

In all the modules, we followed the Software Life Cycle Standard ISO 12207. The participants selected one process to improve from the standard. Furthermore, we presented Software Best Practices and selected Process Improvement Experiments from other European companies that we thought would be a useful reference. Thus we tried to knot together the ISO 12207 Standard, known techniques to improve the processes as well as practical experience from other companies. The processes selected by the participants were the acquisition process, the documentation process, the management process and the development process.

We tried to pilot each module with one company beforehand and this worked well in the beginning. Later it was more difficult to get commitment from the pilot company. This was also the case with the others. When they were at the point of starting the improvement experiment itself, commitment seemed to lack. This may have been due to lack of human resources. We suspect however that it may have been because we didn't present them with an improvement plan that was strong enough. It seemed important to the companies, both technical and management people, to know beforehand how much time and effort an improvement experiment would take. During an interview, some participants said that the process improvement experiment took more time than they had expected.

At the beginning of our club, we decided to omit a module on assessment. The main reason was that we suspected that this would take too long. Also we estimated that most of the companies were of similar maturity and would easily see what process they should try to improve. Although we recognise that an assessment module is useful, as we have seen in courses on SPICE, we are still of the opinion that it can unduly lengthen the process improvement process.

Four participants agreed to be interviewed after the conclusion of the mini-PIE club. The interview is based on a questionnaire that consists of three parts. The first part contains eight basic questions on management and the type of baseline project. The second one contains questions on successes of the experiment and the final part contains eight questions on obstacles. We will summarise the findings around three factors that are human resources, impact of the experiment and external support available.

Those who participated both directly in the club and indirectly in the companies gained more knowledge, enthusiasm and experience in the area of software process improvement. Awareness has been raised within the companies. Members of development teams were informed on the experiments but follow-ups and information for new members lacked. This last point is especially important since all participants said that there had been changes of personnel during the project. Members were motivated to participate but sometimes they were busy with other projects.

Turning to the impact of the experiments, two of the four participants said that it was difficult to measure it. Only one participant was able to finish the experiment within the club so direct impact was not measurable. Two of the participants were able to measure during the experiments. The participants had expectations that the customers would be more satisfied, planning would be improved and cost would be lowered. Two of them said that they thought that there would be fewer errors, fewer complaints and one said that testing time would shorten. The participants had difficulties with the metrics part of the project since they said that they didn't have any baseline data. They did not feel that there was resistance to perform the measurements or resistance to make changes. Two of the participants thought that the experiment took time away from other projects. We need to emphasise that the companies were not able to incorporate a consistent metrics programme, and the above findings are more based on expectations than experience.

The last factor we tried to evaluate was the support available during the experiment. All the participants said they received the support they asked for. They also used the web quite a lot. They benefited from meeting other companies and exchanging ideas. Two of the participants said they would like more contact with people with practical experience in doing software process improvements.

We think that the club can be improved by taking the following actions:

- Make a strong improvement plan with an estimate of resources and duration of improvement experiments. This will hopefully lead to commitment.
- Consultants meet with individual companies at their site at least three times: once in the beginning, once while making the improvement plan, including selecting the metrics and once while doing the improvements.
- Link techniques and best practices to processes of ISO 12207. A model of how this can be done is the SPIRE handbook [6] for the SPICE standard.
- Add a separate module on ISO 12207 after the second module in table 2.
- Make available more reading material that is linked to the individual steps of the experiment. A draft of this appears in a handbook (M3) [7]
- A presentation or a visit from a local company that has already done a process improvement experiment.

Individual Short Workshops

After our experience with the mini-PIE club, we wanted to reach more companies and try to raise awareness of SPI more broadly. Therefore, instead of running a relatively closed mini-PIE club we started to offer individual half-day workshops. Each workshop was focused on a specific theme but they all had a similar structure. After an introduction, the participants were asked to answer a questionnaire that helped them to assess their current status of software development with regard to the specific theme. At the end of the workshop the participants were given a home-assignment to try out the methodology and were invited to come to an informal session two weeks later to discuss their experiences. The themes that were covered in as many workshops are listed in Table 2.

Workshop name
Inspections
Configuration Management
Estimating Size of Software

Table 2 Workshops

These workshops were well attended. A total of 35 persons from 15 companies attended the workshops. In some cases a person participated in more than one workshop. We sent out a questionnaire by e-mail to 29 participants and eleven of those answered. We had only a few questions and have summarised the results in Table 3. The result from the first question indicates that there was clear benefit from the workshops and question two indicates that they also attract those that haven't started yet. Those that haven't started improvements gave various explanations: some said that they are trying to decide on the approach, some that they will start when they get a chance and finally some said that co-workers had started work of this kind in the company. When asked about their expectations of gains by quality and process improvement, the participants gave the following answers:

- Better planning
- Savings in development and maintenance
- More focused development process and better error reporting and fixing
- Software with less errors, but care has to be taken not to be bureaucratic
- Increase in reuse and faster development
- Overall better development processes, better products, and increased customer satisfaction.

Question	**Yes**	**No**	**Don't know**
After attending the workshop are you *more* interested in quality and process	8	3	

improvement in software development			
Have you started quality improvements in your company?	6	5	
Do you think that management is ready to commit to improvements?	9	1	
Do you think that the employees are ready for quality and process improvements?	10	0	2

Table 3 Results from questionnaire

It is our view that the short workshops are flexible, easy to manage, likely to attract a broader audience and are of low cost to the participants. The organisers select the themes and only those companies will participate that think they need to improve in the theme's area. We expect that their impact in terms of improvements is less than for those participants attending the mini-PIE club. This has however not been proven. Based on our experience and the evaluation, the workshops can be improved as follows:

- Participants should be contacted after the workshop to help with the homework assignments. The goal is to increase the turnout at the follow-up meeting
- The workshop should directly link to one process in ISO 12207. This would help to put the improvement into perspective and help participants onward on the road to further improvements

Courses

The ESPICE project has organised several courses in the area of software process improvement. Experienced instructors have taught these courses. The Continuing Education Institute of University of Iceland has marketed the courses that have been offered at their facilities. The courses are listed in Table 4.

Course name
ISO 15504, SPICE
PROBE - the acquisition process
SQUID - metrics workshop

Table 4 Courses

People attend the courses in order to gain more knowledge in general but only some will apply the methodology in their companies. No specific evaluation was done for this support. The Continuing Education Institute asks the students to evaluate the courses. Although it is useful for people to advance their knowledge both teachers and the institute could present the courses as a part of an improvement/consulting package.
The improvements we foresee are:

- The students of the courses are offered a refresher in three to six months. At that time they evaluate whether they can apply the methodology in their working environment.

Tutors

The previous three services were the main ones offered by the ESPICE project. A fourth type of service we experimented with was tutoring at individual companies. In this case a student in Computer Science at the University of Iceland tutored a company while it incorporated inspections of programs, database and hardware design. This service has been advertised to students as a possible project but it has not been advertised within companies. This particular project was very successful. In improvement projects we are often worried that employees of the company do not have enough time to do the improvements. The tutor's role is to support with process definitions, metrics and best practices, provide educational material and last but not least define milestones and put pressure on technical and management people of the company to finish the process improvement experiment. This type of service is the most expensive one, since the company has its own tutor, although in our case it was free of charge since it was a student project. The technical people at the company had a good background in quality improvements and were very open to this kind of work. However they didn't have any experience in performing process improvement experiments and thus benefited from having a tutor. The experience of the SCATE project is similar, but their experience is that it is important that a trained SPI champion be within each company.

Other services

We have offered other services. First, we have distributed the Software in Focus newsletter (www.cse.dcu.ie/swinfocus) but have not evaluated the benefit to readers. Second, we have maintained a web-site that has supported the mini-PIE club and the workshops. We have gathered statistics on its usage and have been pleased with the reception. The members of the mini-PIE club have also indicated that they have used it. More effort should be spent in adding relevant information to it. Initially, we foresaw a help desk where people could call or e-mail to ask for information. This has not been used at all. We think that if people don't see the information on our web they will probably not ask for it but try to look somewhere else.

Comparison of techniques

In order to give an overview of the characteristics of the techniques, we look at some attributes for the different approaches. First we summarise the attributes in Table 5 and then discuss each technique individually. Before moving on we describe the meaning of the characteristics:

- Start-up time is the customer's preparation time before receiving the service
- Manpower is the effort in man-hours or man-months the employees spend on the service
- Impact is a metric on how much software processes are expected to improve
- Awareness shows if the service created an awareness of SPI within the company
- Charge for service gives an idea of the direct fee that the company has to pay for the service

For all the characteristics, we have chosen a three-point scale, i.e. low, medium and high.

	Mini – PIE club	**Work-shops**	**Courses**	**Tutor**	**Web site**	**Help desk**
Start-up time	Medium	Low	Low	Medium	Low	Low
Manpower	High	Medium	Medium	Medium	Low	Low
Impact	Medium	Medium	Medium	High	Low	Low
Awareness	High	Medium	Medium	High	Low	Low
Charge for service	High	Low	Medium	High	Free	Free

Table 5 Characteristics of techniques

Mini-PIE club

The mini-PIE club has a medium start-up time for the companies. We expect that the company needs to carefully consider whether to make a commitment of participating in the club and to do an experiment. Planning the experiment inside the company can take some time depending on the type of experiment. When doing the experiment this approach takes up both resources in overseeing the experiment and in carrying it out. It can also be costly in terms of money if the company decides to buy additional software to aid the process improvement. The impact should be medium at least since so many people are involved in the experiment. The awareness should be high as discussions about the experiment are bound to heighten the awareness level of the employees.

Workshops and Courses

The workshops and courses have a very short start-up time for the companies and they should not need long preparation time. The amount of manpower the company puts into them are in the form of attendance and thus depends entirely on how many people the company is sending to the workshop. We assume that a company sends at least two employees that act as change agents in the company and thus heighten the impact of the workshop and the awareness within the company.

Tutors

Provided a company gets a tutor outside the company, the start up time is medium. Otherwise it is high since a tutor needs to be trained. The manpower is also medium. Impact should be high as it is a customised for the company and the awareness is high since the tutor should be visible and an experiment is performed.

Web site

The web site approach has low start-up time and manpower. The employee is searching for resources on web-site outside the company. Impact and awareness is expected to be low as this is on an individual basis as an ad-hoc approach. A different approach can be to develop an internal web site of SPI resources. This takes more manpower but at the same time results in improved impact and awareness.

Help desk

The help desk is deemed low for all characteristics. A person places ad-hoc queries to the help desk.

Remarks on comparison

Obviously, it is rather difficult to characterise the techniques quantitatively as we have done. It is however necessary in order to compare the techniques. The characterisation may be useful when we ask clients to evaluate the services. For example, we can ask the following questions: What can we do to increase the impact of the courses and workshops? Can our suggestions of visits after a workshop or a refresher a few months after the course bring the impact to High? Finally, it will be useful to add other characteristics to the comparison, such as the revenue that the services create for the service provider.

Selecting a suitable technique

When a customer walks in the door of the service provider we visualise that we can assess his needs and recommend services. Those recommendations can be based on the table of characteristics presented previously in the paper. To summarise we can also give the following recommendations:

- Courses fulfil the need for basic software engineering training.
- Individual workshops are for those who are not committed to carry out extensive improvement experiments but want to start out with simple improvements. These are also meant to increase the customer base of the service provider.
- The PIE club that runs over a period of six to nine months is for those who are willing to make an

improvement plan and devote resources to software process improvements.
- A tutor can benefit those that do not have adequate human resources and have difficulties in staying committed.

The goal of the web site is to support the above services but it can also increase the customer base by having a subscription service or publishing a newsletter.

Although we have deemed the help desk to be unsuccessful, this may change when customers builds better rapport with the service provider or the latter has higher credibility.

Suggestions for other techniques

The list of services we offer is by no means exhaustive. Initially, we had a smaller set of services but have adapted the offering according to our experience and needs of our clients. Consultants frequently provide other services such as assessments that we did not offer. In this section we discuss other techniques. The additional new approaches we suggest are based on the following observations:

- Technical people change jobs frequently
- People like to meet and share their experiences
- The success of SPI depends on two factors: the education or training of the technical personnel and the commitment of management

We acknowledge that it is very difficult to improve or define software processes without having commitment from management. There is very much demand for human resources in the IT (Information Technology) industry and therefore technical people often set certain conditions. One of them is to be able to receive continuous education. Another is to work according to defined software processes. A person can thus market itself as an SPI champion or as having an SPI certificate. An SPI certificate is just like an assessor certificate. It proves that you have done one software process improvement experiment that has produced results. Companies often have well educated personnel. In general they may have good technical skills to apply individual techniques, like review or configuration management or testing. However, they may not have the confidence to deploy the techniques and measure the results in an experiment. Consumers, i.e. the users and buyers are often considered the driving force of improving quality. We suggest that personnel recruitment should be another reason for adopting standard software lifecycle processes. Just as a company thinks it is important to have a policy on quality, have a visible quality manager, have a maturity level, or follow a quality handbook, a company can also make visible the number of personnel with an SPI certificate.
A second technique that is based on the first and second observation above, is the one of an open process. Software processes are tailored to a company's need in an open manner. Company does not keep its process private but shares it with other companies. The approach is analogous to the open model of software development but the subject is a software process instead of the software itself. This technique is described in detail in [8].

Conclusion

We have compared techniques to support small companies in improving software processes. We think that the method presented in this paper may be useful to others serving companies in a similar environment as we do. We summarise the method by listing its steps:

1. Analysis of the needs of our target customers
2. Description of the services provided
3. Evaluation of the services

4. Suggestions for improvements of services
5. Comparison of techniques - useful both to clients when they select a service and to the service provider as a basis for improvements of services

In retrospect, we have applied a *service* process improvement. We have carried out steps one and three above in part with interviews with the customers but in part they are our own observations. Both of these steps need to be ongoing and the questions that are placed need to be modified in order to capture the needs and opinions of the customers in a better way.
The lessons we have learnt from offering the service are stated as suggestions for improvements in the sections on individual services. We can summarise them in the following list:

- A customer needs to make an *improvement plan* where he states his goals and the required resources
- *Experience exchange* between companies should be supported further, e.g. by allowing them to share examples of tailored processes, exchange experiences on tools and methods
- Generic services are good but need to be followed up with *consulting* to the companies
- The web should support the services and be *task-oriented*
- A service provider should offer a *range of services*, suitable for different needs of the customers.

Acknowledgement

The authors wish to thank Anita Björk Lund, currently of Iceland Telecom for carrying out the ESPICE survey in 1999. The authors also wish to thank the co-operation of those companies that answered the surveys and the interviews. The ESPICE project is funded by ESSI, European Software and Systems Initiative of the 4th RTD framework of the European Commission.

References

[1] Statistics Iceland. http://www.statice.is/

[2] The Association of Industry, www.ut.is.

[3] Iceland Review, Iceland's Software Industry '98-99, Iceland Business, Special Edition 1998-99, Iceland Review. http://www.icenews.is/

[4] Páll Ásgeir Ásgeirsson, *Litlir risar*, Frjáls verslun, 3. Tbl 1998. An article about the Icelandic software industry.

[5] Innkaupahandbók um upplýsingatækni, RUT nefndin 1998. Guidelines for IT procurements.

[6] The SPIRE Handbook, The SPIRE Project Team, 1998 The European Community, ISBN 1-874303-02-9

[7] Lund, A. B., Hvannberg, E. T. A guide for mini assessment and software process improvement.

[8] Hvannberg, E. T., Thorbergsson, H. An Open Model for Tailoring Software Processes, in Conference Proceedings, Views of Software Development in theh New Millennium, Reykjavik, Iceland, August 2000

Ebba Thora Hvannberg

Ebba Thora Hvannberg is an associate professor in the Department of Computer Science and Systems

Engineering Laboratory. She is the project manager of the ESPICE (ESSI PIE Node in Iceland) project. Her main research interests are software engineering, human computer interaction and distributed services. Ebba Thora teaches courses on Operating Systems, Human Computer Interaction, Formal Methods in Software Development and Distributed Processing. She has participated in numerous international projects such as NFMM (Nordic Fisheries Management Model), AMUSE (Advanced Multimedia Services for Residential users), Distance Education, and Early Warning Systems of Natural Hazards. Ebba Thora received a Ph.D. degree from Rensselaer Polytechnic Institute in Troy, New York in 1988. Ebba Thora can be reached via e-mail at ebba@hi.is

Helgi Thorbergsson received his doctorate from Rensselaer Polytechnic Institute in 1990. He has an extensive experience in the local software industry where he worked until 1997. He has been an associate professor of computer science at the University of Iceland since 1997. His main research interests are in software design, software engineering and distributed databases. Helgi can be reached at hetho@hi.is

Gudrun Agusta Johannsdottir works as a research manager at OZ.COM. Special fields of interests are quality management and information management. She is an M.Sc. student in the Department of Computer Science. She can be reached at agustaj@oz.com

University of Iceland

The University of Iceland (http://www.hi.is/HIHome.html) is a state university founded in 1911. During its first year of operation 45 students were enrolled. Today, the University of Iceland serves a nation of approximately 270,000 people and provides instruction for some 6,700 students studying in nine faculties. The University of Iceland offers opportunities for study and research in more than 50 degree programmes in the humanities, natural sciences and social sciences, and in professional fields such as theology, law, business, medicine, dentistry and engineering. The Systems Engineering Laboratory conducts research in the area of control systems engineering and computer science. It has participated in many projects in these areas, both domestic and international ones, co-operating with other universities and companies in the air transportation, ferrous silicon, aluminium, telecommunication and IT industrial sectors.

Session 14 - SPI and Requirements

Session Chair: John Elliott, SEC, DERA

Linguistic methods of Requirements Engineering (NLP)

The incorporation of linguistic methods (NLP) in the investigation and quality assurance of requirements – an experience report from industrial practice

Christine Rupp
SOPHIST GmbH, Nürnberg

SOPHIST GmbH

Key Words

requirements, requirements analysis, requirements engineering, linguistic methods, software analysis, specification, specification document, Neuro Linguistic Programming (NLP), Meta-model of language.

Summary

In order to avoid errors in the analysis phase of software engineering, and also to further integrate clients and users needs in the analysis phase, IRE 9000 (Integrated Requirements Engineering) intends to use linguistic techniques in order to investigate and review requirements. This is not another approach to formally describe requirements but a method to work on natural language requirements directly at the specification document level. The overall goal of object engineering is to produce high-quality, legally binding requirements in natural language as a basis for systems development.
SOPHIST Ltd. transferred research from other sciences, i.e. linguistics and psychology, to computer science, and the result is an easy-to-use set of tools to find and avoid ambiguous, incomplete and inconsistent requirements. This paper will examine the various linguistic phenomena that underlie errors found in requirements. Furthermore, the paper will show how these errors can be found and corrected.

Motivation

Many studies show that the majority of all defects in software products originate in the analysis phase. Mistakes that originate in the early phases of software development cause a “snowball effect” which propagates many more mistakes into later phases of development where it is much harder to remedy. Sometimes the situation is beyond repair. Approximately 60% of all serious mistakes in programs can be traced back to deficiencies in the analysis phase.
In order to control exploding costs and development time, the root causes that lie in the analysis phase must be removed before they can do serious damage. Improvements at the analysis stage level are more effective than in the design phase and incomparably better than in the implementation phase. The earlier an error is found and fixed, the less damage it can do to later development and the better the important development parameters of cost and time can be controlled. The first phase in the development of a project is decisive; it usually decides the success or failure of the project.

Requirements Engineering

Requirements Engineering (RE) concerns itself with the discovery, documentation, organization, review and administration of requirements for a software product. From an analyst‘s point of view, a typical scenario is as follows:
A semi-detailed specification document with excessively large sections and vague formulations of requests is submitted in natural language (prose) to the client‘s analyst. The analyst may then perhaps transform this specification document into a semi-formal representation such as a SA-, SADT- or OO model. On the basis of this representation, the analyst researches further and adds important points that he determines through dialog with the client or user(s). The original specification document loses all relevance as the model of the analyst shapes all new requirements, hence the original specification document becomes useless.
The result of this quite everyday scenario opposes the thought and purpose of the original representation because the analysis model should be understandable for all participants in the process and not merely from the analyst’s point of view. A clear representation enables the user to follow what the analyst models. The user can immediately “change course” if he notices that the model has gaps.

Perhaps some points are missing or are not adequately addressed. Lastly, (a) the user is more certain that he will receive what he wants and (b) in case of a dispute regarding system functionality (keyword acceptance), he is not standing in legal deep yogurt.
For reasons mentioned previously, natural language requirements (prose requirements) must now be given the primary status, which is opposite the previous convention. This does not mean that the current (semi-) formal representations of requirements are incorrect and useless; object engineering requires the translation of prose requests into an integration model and a simulation model. Rather, the representation shall be in natural language and the subsequent work with the natural language requirements is an integral component of the analysis process.

General consequences for the representation of requirements

In the development of a system, two different points of view emerge. On the one hand there are the analysts, designers and implementers and on the other hand there are the clients and users.
These two often-diverging points of view must be integrated in the development process. Here the transposition of requirements into an executable program and therefore the model of the future system must include the totality of all requirements and the model's representation must be formal. The analyst and designer pursue this overall goal as if they were the system implementers and therefore they must keep this "prime directive" constantly in mind. The user however is much more interested in a delivered system that corresponds to his ideas. If the program does not correspond to the customer's expectations and mental "grand plan" complaints will follow.
The criteria for the model, which includes the totality of the requirements, are essentially derived from these two previously mentioned points of view. To further this goal, the requirement sentences should be --

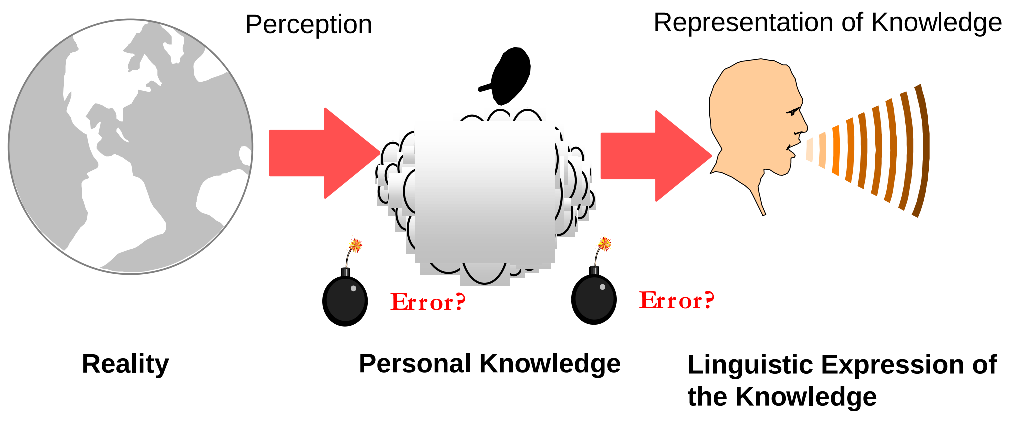
- unambiguous
- complete
- logically consistent and
- testable

These points are the basic requirements of a formal description and are also the basis of the legal obligation or contract. Therefore these requirements are quite significant indeed which requires one more point and that is that they are

- clear

which, besides the already named reasons, greatly facilitates the understandability and hence acceptability of the requirements to the user.

Fig.CRUPP.1 : Reality, personal Knowledge, expression of the Knowledge


Perception
Representation of Knowledge
Error?
Error?
Reality
Personal Knowledge
Linguistic Expression of the Knowledge

Consequences for the representation of requirements in prose

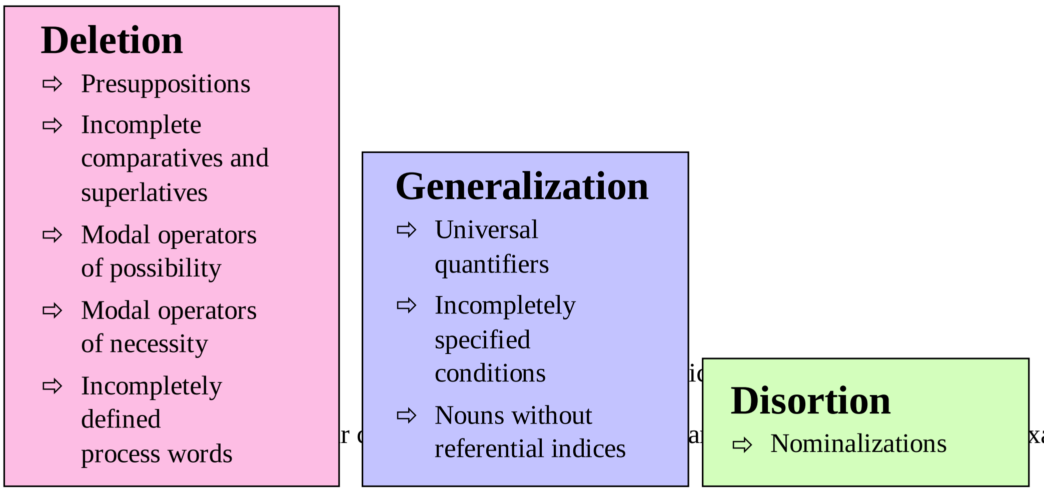
In a natural language representation, there are a few problems that lie in the nature of natural language that can make the necessary formal model an informal model. Remarks in prose are often (usually!) broadly interpretable, so previously mentioned criteria, unambiguity, completeness and logical consistency, seem to be impossible goals! Moreover, prose is generally not suitable for describing complex content structure while maintaining clarity of representation.
To this end, a set of rules for writing and checking of prose requirements would be very expedient and helpful in removing or minimizing the deficiencies and ambiguities of natural language. The goal of an analysis process on the basis of prose requirements must be the named formal criteria of legal obligation and intelligibility for all participants. Happily, such a set of rules exists and the rules can be followed. Therefore they should be put into practice.

Method of application in the analysis of prose requirements

The goal of a systematic language analysis is to uncover formulations that are stamped by subjective experience and to replace these subjective formulations with clear, objective formulations. A method of action is only possible at this point because natural language does possess an independent system of analysis. The discoverers of this system are linguistic scientists who have already examined and defined models of language, parts of which may be useful.
The father of the theory of a systematic construction of language is Noam Chomsky, founder of generative transformational linguistics. The results of his theory make it possible to build a definite sentence by applying grammar rules to any language component, be it spoken or written. Chomsky’s theory has undergone certain expansions and alterations since its first publication because the theory didn’t adequately explain certain aspects of language.
The method of analysis of prose requirements is essentially the application of the improved theory of transformational grammar. Therefore, one must, by means of generative rules, subject sentences to transformations. Ultimately the product is a grammatically correct and exactly defined sentence.
More recently, the basic science from linguistics has been applied to a model of human communication and mode of expression. This has produced a set of psychology rules or transformations, which enable an exact meaning of spoken or written sentences to be found via the application of transformations upon the spoken or written sentences. Correctly applied, the underlying meaning of the communication is revealed. Avenues of further questioning and investigation are also evidenced. The original set of rules are enumerated in broad strokes in the book, “The Structure of Magic I,” by R. Bandler and J. Grinder, the creators of therapy-based Neuro Linguistic Programming (NLP).
The firm SOPHIST Ltd. has similarly applied models of language upon areas of Computer Science. The result has been the creation and testing of simple, easy to use rules for the creation and quality assurance of prose requirements in project development.
Through the use of these flexible and adaptable rules, a systematic mode and manner for the review of natural language requirements formulated in natural language is applicable. This is particularly true in regards to eliminating requirements, which contain ambiguous, incomplete and inconsistent statements as are often found in requirements documents. Finally, this method can and should be utilized in the writing of requirement documents so that errors or ambiguities never even “see the light of day.”

The linguistic methods in the investigation and quality assurance of requirements

In the following, the most important principles of application will be described. The chart below exhibits the most frequently observed language phenomenon from a linguistic point of view. Due to extreme time limitations for the presentation, only one representative sample of each type of defect will be exhibited.


Deletion
⇨ Presuppositions
⇨ Incomplete comparatives and superlatives
⇨ Modal operators of possibility
⇨ Modal operators of necessity
⇨ Incompletely defined process words
Generalization
⇨ Universal quantifiers
⇨ Incompletely specified conditions
⇨ Nouns without referential indices
Disortion
⇨ Nominalizations

Deletion

Deletion is a process through which we turn our attention towards selectively determined dimensions of experience and exclude others for simplification. In other words, the process of deletion reduces the world into smaller pieces, which can be more easily handled. This reduction can in certain contexts be quite helpful, but in the area of requirement definitions for a software system, we must know exactly what information has been lost.
The following examples exhibit various types of deletions and give an insight into the spectrum of possible representations from deletion transformations.

Presuppositions

Presuppositions are statements that must be true so that a statement has any meaning. This form of deletion is frequently found in requirement documents. Implicit assumptions are also presuppositions. For example, so that the sentence

When the altitude is less than the minimum height, an alarm should be set.

has any meaning, the statement

There is a minimal height.

must be true.
Presuppositions or implicit assumptions must be made explicit in requirement documents in order to be meaningfully complete. Presuppositions originate frequently through an omission by the author of a requirement because the presuppositions are either so obvious to the author that he or she doesn't consider it noteworthy or the author is not even aware that there is an implicit assumption.

Incomplete comparatives and superlatives

In comparison statements like

This information can also be accessed from slower storage media.

you must place yourself in the question's frame of reference. Slower than what? How slow is slow?

In

The corrections should be easily modifiable.

the question remains what exactly is easily?
A comparative or superlative always requires a reference point to be completely defined. Furthermore, the unit of measure (ex. Meter, Second) and the tolerance (ex. +/- 0.1 meter, +/- 0.0001 seconds) must be declared.

Incomplete Process Words

Process words are those words in a sentence that describe a process. They must not necessarily be verbs, as adjectives and nouns can also play this role in a sentence (see also Nominalization). In order to be complete, a process word usually requires more arguments or perhaps even an entire noun phrase to be completely declared. Consider the following:

The system should report data loss.

The process word "report" is completely defined only if the following questions are answered: Who reports? What is reported? How or in what manner is the information reported? To where or to whom is it to be reported? When is it reported? For how long a time is it reported?

The utilization of internal system resources should be monitored.

In this statement "utilization" is in its noun form as a process word. To be completely defined, a process word must first answer the following questions: Who or what exactly is utilized? How is it utilized?
Furthermore, the process word "monitored" is referential. The statement is defined completely only if the following questions are clarified: Who monitors? What is monitored? How or in which manner is it monitored?

Generalizations

The human ability to experience through generalization is a process that is both meaningful and necessary for survival. Through generalization, we are able to transfer an experience to related contexts. It is very important to consider the related context in which the experience is transferred, particularly in regards to the information that may have been omitted in the application of the generalization.
Through the process of generalization requirements are often made which seem to apply to a large or entire part of a system. For other parts of the system, these requirements can be very false indeed, whereas a correctly defined requirement would actually apply to a smaller piece of the system in order to have accurate scope and correct meaning.
Typical for the process of generalization is the suppression and omission of both special and error cases respectively. In the following, the most frequently seen variations of generalization are reviewed.

Universal Quantifiers

Universal quantifiers are parts of statements, which are broadly applied to all incidences of occurrence. Linguistic representatives of this group are concepts like, "never," "always," "no," "every," and "all." The danger with the application of universal quantifiers is that often the specified behavior does not truly apply to all referenced objects in the group or set. The universal grouping usually contains one or more special or exception cases, which the universal quantifier references, but for those cases, the specified behavior is false.

Each signal shall additionally be labeled with a time stamp

On the basis of the keyword, "each," a question is immediately evidenced. Really each/every signal? Or are there perhaps one or more cases in which the time stamp is not required?

It is critical with this type of generalization to immediately define the range of applicability so that no possibilities and occurrences of applicability are left out. Special cases must also be defined in the generalization process.

Incompletely specified conditions

Incompletely specified conditions are another indicator for a possible information loss through a generalization. The usual representations, among others, are, “if”, “then,” “in case of,” and “depending on.”

> *If data X is needed by interface Y, then the time for a response shall be under 0.5 seconds.*

The begged question is this: which time requirements exist in the case that data X is not requested by the Y interface?

Nouns without reference index

Noun arguments without (clear) reference indices are another key type of information loss. This means that a noun exists in a statement but the noun is not sufficiently specified for clarity of the sentence.

> *The message shall be displayed at the working position.*

The nouns of this sentence, message and working position, give no hint as to what exactly they refer. Therefore at least two questions pose themselves:
Which message? Which/whose working position?

Distortion

The distortion transformation appears almost exclusively in form of a nominalization.

Nominalization

A nominalization occurs when a process is reformulated into an event. A nominalization may change the meaning of a statement. A nominalization may also cause important information regarding a process to be lost. Linguistically one recognizes a nominalization as a process word (verb or predicate) that has been molded into an event word (noun or argument).

> ***Data loss** shall be recognized and reported.*

The process behind the noun *data loss* actually consists of:

> *Data is being lost.*

So this sentence has the related questions: Which data is being lost? How is the data being lost? How can the loss of data be recognized?

Results and conclusions

The linguistic methods of Requirements Engineering introduced here have already been successfully applied in several large industrial projects. In each case, the defined procedural guidelines of object engineering played a crucial role for the success of the project. The previously mentioned success of the linguists and psychologists suggests an easy and useful transfer of proven and utilizable tools onto the software development process -- particularly in the engineering of prose requirements.

When picking methods from the methods kit object engineering it is hardly possible to omit the component linguistic methods in contrast to the possible elimination of the integration and simulation model components. Object engineering forms the basis for further work in the software development process with prose requirements, which are the essential foundation for continued development, as previously demonstrated.

Hence, with some concessions, it is possible that the scientifically necessary formal model, which is formed by the entirety of all requirements, can be represented informally with the assistance of linguistic methods. So the door is then opened such that the client and user may, for the first time, directly review and criticize the working description of the problem.

References

Foundations for this paper follow:

- Software Engineering Handbook of the Deutsche Flugsicherung Ltd., produced by SOPHIST Ltd.
- Description of the action model Objects 9000, decisive production from C. Rupp, SOPHIST Ltd.
- Manuscripts for the book, “Linguistic Methods of Requirements Engineering”, C. Rupp, SOPHIST Ltd.
- Training course, “Requirements Engineering — Linguistic Methods”, SOPHIST Ltd.

Profile and address of the author

Christine Rupp is CEO and owner of SOPHIST Ltd. and has 7 years of extensive experience in the area of software analysis methods. She has led and supported projects with organizations including the Deutsche Flugsicherung Ltd. (German Flight Control Administration), Eurocontrol, Deutsche Post AG, Daimler Benz AG, RWE DEA and Siemens AG.

Address:

SOPHIST Ltd.

SOPHIST Gesellschaft für innovative Software-Entwicklung mbH

Vordere Cramergasse 11-13

90478 Nürnberg, Germany

Tel:	**+49 (0)911/ 40 900-0**
Fax:	**+49 (0)911/ 49 900-99**
Email:	chris.rupp@sophist.de
Web:	http://www.sophist.de

Improving Requirements Engineering Capabilities

Dr. Shai Koenig
ECI Telecom Ltd., Israel and the Open University of Israel
Izik Zelovich
ECI Telecom Ltd., Israel

Introduction

This report describes a process improvement experiment [PIE] whose goal was to improve the company's requirements specification and management capabilities. The PIE was conducted by ECI Telecom Ltd. under contract to the ESPRIT/ESSI Program of the European Commission [ESSI PIE 27582, ARMT]. ECI Telecom is Israel's largest supplier of telecommunications products, selling to over 600 customers in over 145 countries. ECI Telecom is involved in the specification, development, production, sales and service of a wide range of products and systems for the telecommunications market, including transport network systems, access systems, private exchanges, computer telephony integration and wireless access systems. In total, the company employs over 5000 workers, of which over 1000 are systems and software engineers.

The objective of our PIE's was to concurrently improve two core requirements engineering capabilities - the requirements specification capability and the requirements management capability. We view these as two relatively orthogonal, but complementary axes. Along the requirements specification axis, the PIE's objective was to infuse the newer Use Case/Scenario approach to expressing functional requirements. The use case/scenario approach represents a major paradigm shift from the commonly-used, traditional, prose-based "the system shall/the software will" model for defining requirements. Along the requirements management axis, the PIE's objective was to replace a document-centric approach, in which requirements information is recorded and managed using conventional documents and document preparation tools, with a database-centric approach, [in which requirement information is directly entered into a structured, multi-user database, requirements documents are generated automatically from the contents of the database, and requirements information is managed using advanced database capabilities.

This report describes the PIE's strategy, high level and detailed objectives, the way the PIE planned to achieve these objectives, the implementation of the PIE plan, the results of the PIE and the lessons learned.

Some Observations on Improving Requirements Engineering Capabilities

The ability to elicit, specify and manage system and software requirements is a core capability needed to successfully compete in today’s highly dynamic telecommunications products and systems marketplace. Nonetheless improvements in requirements engineering activities lag behind improvements in other software development activities such as design, construction, and testing, which seem to be more amenable to process improvement and automation. The direct objective of our PIE is to improve this requirements engineering capability via the introduction of more advanced requirements specification methodologies supported by automated requirements management tools. Improvements in these core capabilities will facilitate the development of our products/systems within defined schedule, cost and quality goals. Our PIE attempts to make improvements along two relatively independent axes: the requirements specification axis and the requirements management axis.

The Requirements Specification Axis

There is a wide spectrum of ways to express technical requirements. On one extreme are the unstructured, natural language approaches. “The system shall / the software will” style of requirements specification has characterized most of the industrial requirements engineering community over the last 30 years. Although informal prose is easily understood by a wide range of requirements specification “customers”, informal prose may also be easily misunderstood and suffers from problems such as ambiguity, incompleteness and inconsistency. On the other extreme are formal methods based on formal languages such as Z, VDM, Algebraic Specifications and Statecharts [1,2,3]. Although requirements specifications based on formal approaches may be more precise and less prone to the problems encountered with informal prose-based methods, they require a high-level of expertise both to produce as well as to read and understand. Most requirements specification “suppliers” and “customers” do not have this requisite expertise. Between these two extremes lie a number of approaches based on the use of structured prose and semi-formal diagramming languages such as data flow diagrams [4,11], entity relationship modeling [4,11], and the more recent use case approaches introduced by [5].

Like most telecommunications companies, we, at ECI, have traditionally used the unstructured, natural language “the system shall / the software will” approaches to requirements expression. Along the ***requirements expression axis***, the PIE’s objective is to infuse the newer Use Case/Scenario approach to requirements specification [5,6,7,8,9]. This is a major departure from the more conventional, prose-based “the system shall/the software will” model for defining system and software requirements. The Use Case/Scenario approach that we have adopted seeks to improve requirements specification by providing a more structured, “user”-oriented, transaction-based approach to specifying functional requirements.

The Requirements Management Axis

Once requirements have been specified they must be managed. This involves organizing requirements so that they can be queried and accessed by various stakeholders in various ways; tracing requirements from their antecedents [e.g., marketing requirements] and associating them with other engineering information [e.g., related requirements, test cases]; assigning management attributes such as priority, level of stability, and status; controlling the evolution of requirements; allocating requirements to product releases. Such requirements management is not feasible when requirements specification is conducted in a document-centric manner. In fact, requirements management is just a form of information management, and therefore should be based on information management approaches including database management, client-server architectures and query facilities.

Therefore, along the ***requirements management axis***, the PIE’s objective is to replace a document-centric approach to recording and managing requirements using documents and word processing tools with a database-centric approach, in which requirements are directly entered into a database management system and then managed using the database management system. Documents, when necessary, are generated automatically. Reports may be similarly defined and generated as needed. This affords all the advantages that database management systems provide for storing, viewing and manipulating data, while making the necessary documents available as needed.

Objectives of the PIE

As a result of the introduction of the practices described above we expect, in particular, to observe improvements in four requirements related areas.

Improving the human factors aspects of the requirements specification process

We expected that the proposed improvements in the requirements specification and management capabilities described above should have salutary effects on the human factors aspects of the requirements specification process. In particular, we expected to observe a significant decrease in the effort needed for requirements specification, an improvement in team coordination, an increase in a project's ability to use less experienced staff for requirements specification, and an improvement in the learning curve for new project staff to "enter" the project.

Improving our ability to articulate the requirements

We expected that the proposed improvements in the requirements specification and management capabilities described above should result in significant, user-perceived improvements in the desired attributes of the stated requirements. In particular we expected to observe improvements in the basic attributes of a requirements specification – correctness, lack of ambiguity, completeness, consistency, understandability [by "customers", developers and testers], verifiability, testability, level of detail [10,11].

Improving our ability to manage the articulated requirements

We expected that the proposed improvements in the requirements specification and management capabilities described above should result in significant, user-perceived improvements in the organization and structure of the requirements; the ease with which requirements can be modified; traceability - the ability to relate requirements to other development information [such as other forms of requirements, design decisions or test information]; the accessibility and "view-ability" of requirements information; the ability to characterize requirements in terms of management attributes [e.g., importance, effort required, etc.] and use these attributes to analyze requirements.

Improving the impact of requirements on other development activities

Requirements specification is the basis for subsequent project planning, design and validation testing activities. We expected that the proposed improvements in the requirements specification and management capabilities described above should provide a better basis for feature roll-out planning, for time and cost estimation, and for the ensuing design, test planning, change control and release management activities. If achieved, these technical improvements should contribute to our business goals by leading to better control of complex projects, reduced software development costs and delays, enhanced product quality and improved customer satisfaction.

The PIE Plan and Its Implementation

In this section we describe both the original plan for achieving the PIE's objectives and its implementation. The project was broken down into the following primary phases:

- Methods and Tools – Survey, Study, Selection and Acquisition
- Establish Requirements Engineering Environment
- Internal Training of RM Process
- Requirements Engineering Process & Tool Support
- Measurement Planning and Implementation
- Internal Dissemination

Each of these phases is briefly described in the following subsections in terms of the original plan, its subsequent implementation and problems encountered.

Methods and Tools – Survey, Study, Selection and Acquisition

Phase Plan - This phase of the PIE involved studying requirements specification and management methods and tools, selecting the corresponding methods and tools best suitable for our company and acquiring them. This was to involve an initial survey of existing requirements specification methods, management tools and related processes resulting in a short list of tool candidates; a comparative and detailed study of the tools and related processes on the short list of candidates; and final selection and acquisition of the selected methods and tools.

Phase Implementation - Prior to the onset of the PIE, we had already been experimenting with use case/scenario approaches to specifying requirements. Further investigation [5,6,7,8,9] convinced us that the use case/scenario approach should serve as the basis for the requirements engineering improvements to be instituted by our PIE. We adopted the newly emerging UML standard [7,8] for use cases and scenarios as the basic method for specifying functional requirements. We then set about investigating the available commercial tools for performing requirements specification and management that would support the use case/scenario model. In the first stage of the evaluation, we made a cursory survey of existing automated requirements engineering tools. This included perusing the relevant literature, searching related internet sites, attending presentations at a locally held conference and visiting companies in Israel that had begun using such environments. This had two purposes. One, to aid us in understanding the various features and capabilities of such environments. This helped us in articulating our list of selection criteria. Second, to establish a short list of candidates for further, detailed evaluation. We considered DOORS, RTM, Rational's Requisite Pro and the use of MS Access together with Rational's Analyst Studio Suite.

From a technical viewpoint RTM was our selection of choice. It provided the best combination of support for requirements management and for expressing requirements based on the use case/scenario method. However RTM, like DOORS, had no representation in Israel, and we felt that such support would be critical to our success. Requisite Pro, which did have local support, was judged not to have the semantic richness needed to support the use case/scenario approach that we had selected. We therefore chose to use MS Access for entering, storing and viewing requirements information. We supplemented this with Rational's Analyst Studio or iLogix's Rhapsody for support of UML Use Case diagrams, interaction diagrams, and when necessary Class diagrams. The added advantage of choosing Rational or iLogix tools is the continuity that they provide to subsequent design activities. The selection criteria as well as the results of this investigation were recorded in our *Requirements Management Environment Selection Report* [12].

This phase took much longer than we had originally planned. On the one hand, the use case/scenario approach to requirements engineering had not yet stabilized and was still evolving as part of the emerging UML standard [7,8]. On the other hand, existing automated tools for requirements management had not been designed to give direct support for this methodology. This was complicated by the fact that some of the tool vendors had no representation in Israel at the time. As a result, considerably more effort than expected had to be invested in attaining the needed balance between methods, processes and tools that is required for building an effective methodology-driven automated environment.

Establish Requirements Engineering Environment

Phase Plan - The objective of this phase was to establish the necessary requirements engineering environment [consisting of methodology, tools and training resources] for baseline project use. This phase was to involve refining and tailoring the selected requirements methodology, adapting the selected tools to the requirements methodology and preparing training resources.

Phase Implementation - We first established the details of the use case/scenario method to be used. As mentioned earlier, the UML standard was still evolving at that time. We adopted the standard, but the standard left a number of significant issues open. In particular, UML does not specify a specific notation or structure for specifying the details of use cases and textual scenarios. Based on our previous work and the emerging UML standard we defined our Use Case/Scenario data model [see figure 1] and the structure and contents of the requirements specification documents to be automatically generated.

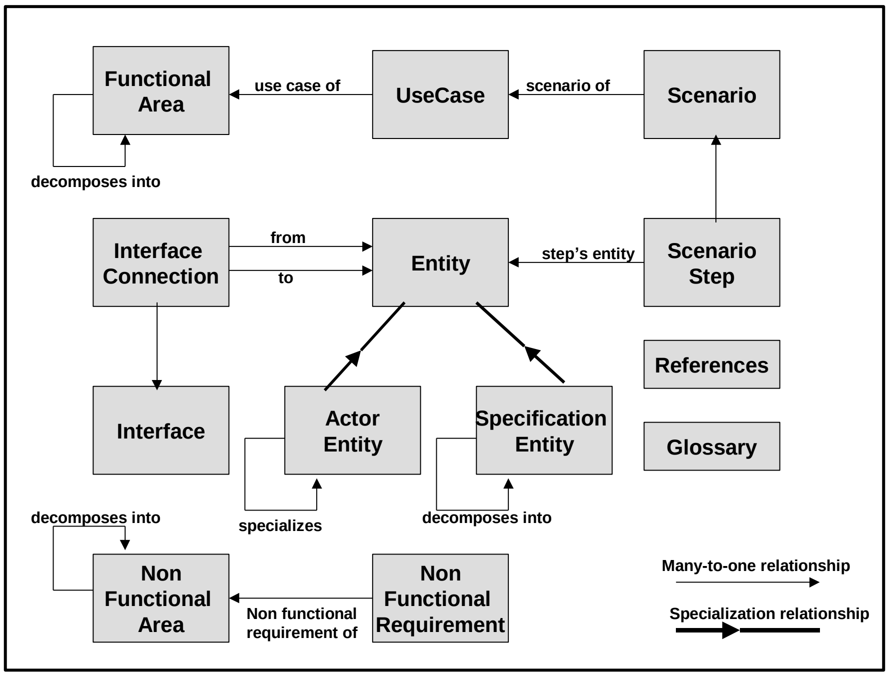


Figure SKOE.1 – Class diagram for the conceptual schema

In particular, the PIE support team learned MS Access and Rational's Analyst Studio and adapted it to our Use Case/Scenario approach to requirements specification and management. This involved

- defining the conceptual schema using class modeling techniques [figure 1]
- defining the logical database schema to be used in the requirements database
- defining the data entry and data view forms needed to support our Use Case/Scenario approach
- writing the scripts to automatically generate the requirements documents from the database contents [see figure 2 for an excerpt of a requirements document that was automatically generated from the project's database].
- integrating the relevant Rational's Analyst Studio components [use case diagrams and sequence diagrams] with the MS Access tool.

We call this collection of tools EIMS [for Engineering Information Management System]. To verify the suitability of our approach, we used EIMS for an internal pilot project prior to releasing it to baseline projects.

A training program consisting of three courses was established. The first course is a short course that presents the fundamentals of the use case/scenario approach to requirements specification, the principles of requirements management and the use of EIMS. This was supplemented by a course in object-oriented analysis and a course on the use of Rational Rose [the part of the Rational Analyst Suite that we are using in the project].

3.1 **FUNCTIONAL AREA**: Alarm indication while config. file running

BRIEF DESCRIPTION: The main target of DNMS application is to give the user a tool to manage different types of equipment. A major part of management process is alarm indication

3.1.1 **USE CASE**: Alarm indication of EMOPS while file running

Use-Case Goal	To display to user alarm indication of EMOPS
Triggering Entity	EMOPS-240
Triggering Event	Alarm changing in EMOPS window
Entity Precondition	EMOPS window is on the desktop running Reflection X
Success Postconditions	Alarm successfully taken from the window and displayed to user
Failed Postconditions	Failure while taking alarm from EMOPS window
Related Use Cases	
Remarks	None

DECOMPOSITION: This use case is decomposed into the following scenarios.
3.1.1.1 **SCENARIO**: EMOPS alarm success

Brief Description	EMOPS current topmost alarm displayed to user
Scenario Type	Primary
Entity Precondition	EMOPS window is on the desktop, launched through Reflection X
Remarks	None

Scenario Table:

Step No.	**Entity**	**Step Description**	**Remarks**
1	EMOPS-240	Changed alarm state	
2	X Sites module	Catch alarm change, update system database, send signal to update view of alarm indication	
3	Alarm indication module	Display bell alarm at toolbar, color icon on GIS map, audible notification	

Figure SKOE.2 – Excerpt from an automatically generated SRS

Internal Training of the Requirements Engineering Process

Phase Plan - The objective of this phase was to train the users in the baseline project in the selected requirement engineering methods and tools and prepare them for performing the requirements specification and management function within the baseline project as defined by the previous phase.

Phase Implementation – The PIE postulated a single, large baseline project. Although we did not include the actual selection of the baseline project within the original PIE process plan, this selection process required significant efforts. It involved identifying candidates for our baseline projects, presenting them with the details of the PIE and trying to persuade them to participate in the PIE. Although most projects expressed interest in the PIE, large projects were not willing to make the necessary commitment to the PIE. Time pressures and the natural reluctance to take risks inherent in introducing new techniques and tools [particularly in a development phase that is viewed by many as a necessary evil to be dispensed with as soon as possible] were the primary reasons cited for this lack of willingness. At this point we decided to adopt another strategy. Instead of one large baseline project, we decided to look for a number of smaller projects. We succeeded in identifying and convincing a number of such projects to commit to the PIE. The main ones are applications based on a CTI [computer telephony integration] Call Center developed by the Business Systems SBU of ECI Telecom. Members of the baseline projects were provided training in accordance with the PIE's training program as described in the previous section.

In retrospect, it was a mistake not to identify and budget baseline project selection as a separate task. This task both consumes effort and duration and clearly lies on the PIE's critical path and therefore should be made explicit. Although not part of the original plan, the decision to use several smaller baseline projects has a number of significant advantages. Foremost is the reduction of the risk of a project being canceled or "jumping ship" in the midst of the PIE. Many projects abandon process improvement efforts during the project's lifecycle. Choosing several baseline projects eliminates this "single point of failure". In addition, having several baseline projects provides several independent assessments of the PIE's results and provides more solid support for the PIE's conclusions.

Process & Tool Support

Phase Plan - Open issues, questions and difficulties invariably accompany the introduction of a new method and its supporting tools. The objective of this phase was to prevent these open issues, questions and difficulties from endangering the success of the PIE by providing on-going, day-to-day methodology and tool support. We budgeted and allocated the resources so that such support would not suffer. In addition, we planned to track problems and enhancement requests using an automated task management system.

Phase Implementation – The PIE team divided this support function into two forms - reactive and proactive. Reactive support involves responding in a timely manner to questions and problems that the project may encounter in the course of their everyday work. Proactive support involves periodic visits to the project staff to assess their progress and identify potential problems even before they are raised. Problems that could not be resolved immediately were recorded in the PIE's task tracking system using iTTS, a corporate integrated task tracking system. In addition, use of EIMS in the baseline projects inevitably leads to suggestions for improvements to both the methodology and tools. These suggestions were similarly recorded and analyzed using iTTS and are serving as the basis for enhanced versions of the EIMS environment. A number of such issues are discussed in the final section of this report.

This phase used significantly less resources than we originally planned. Users did initiate feature enhancements, which had to be analyzed and implemented. In addition, there were a number of bugs in the tool environment that required our intervention, but less than we expected. Regarding the need for on-going support, it seems that the method and supporting tools are quite intuitive and as a result we spent less effort than expected providing methodology support. Invariably, a local "champion" emerged in each baseline project who not only championed the new approach but also served as a first line of support within his respective baseline project.

Measurement Planning and Implementation

Phase Plan – The goal of this phase is twofold - to develop a measurement program for assessing the extent to which the objectives of the PIE have been achieved and implementing that program. This involves developing a measurement approach, collecting data, analyzing the data and summarizing the date.

Phase Implementation – In the course of the PIE, we extended its objectives as described in a previous section. In light of this change, we had to rethink the initial approach to measurement. Following the GQM [Goal-Question-Metric] approach [13], we developed a survey questionnaire that translates the PIE's goals and subgoals [what we have called detailed objectives] to questions. This questionnaire is filled out in the course of personal interviews with baseline project participants and baseline project "customers". For each detailed objective, the questionnaire asks its respondents to rate the importance of the objective and assess the change [i.e., significant improvement, improvement, no change, deterioration, significant deterioration] resulting from the use of EIMS in comparison to the way requirements were specified and managed prior to his use of EIMS. In addition the interviewee is asked to attribute the change [improvement or deterioration] to the new methodology or the new automated tools using a sliding scale. This data will then be analyzed to better understand the importance of the objectives, the extent to which the PIE has or has not achieved them and the main cause of the change. As of the writing of this report, the questionnaire has been developed, reviewed and completed, and the personal interviews have been conducted. Preliminary results based on these interviews are presented in the next section of this report. Detailed results will be presented in a future paper.

Internal Dissemination

Phase Plan – The objective of this phase is to inform the ECI Telecom community regarding the results of the PIE. This includes in-process internal dissemination via presentations at corporate and division levels, a corporate recommendation regarding requirements management technology adoption, lead infusion into other projects (not part of the PIE) via presentations and demos of the baseline project's experiences and results.

Phase Implementation – Although final assessments of the PIE's baseline projects have not been completed, initial user reactions have been quite enthusiastic. As a result we have decided to present our approach to interested parties throughout the company. As of the writing of this report we have made twelve such presentations, including one to the ECI Corporate SW Development Improvement Forum, one at the company's annual technology seminar and the rest to development groups within the company. As our experience with this new approach grows, we continue to modify the presentation and update the relevant ECI groups with the results of the PIE. The following section briefly describes some of the company's future directions regarding the approach that we have introduced in our PIE.

The PIE Results and Lessons Learned

At the time that this report is being written, the assessment phase is being completed. The questionnaire data has been collected and is being analysed and summarised. In the next few paragraphs we shall present a qualitative summary of the findings. Users found the new requirements environment [the use-case/scenario methodology together with the supporting automated tools] to be a significant improvement over their previous approaches. Users rated the EIMS environment as providing either an improvement or significant improvement for the majority of the PIE's evaluation criteria. Moreover, the respondents felt that all the evaluation criteria addressed by the questionnaire were important. Specific lessons learned are presented below.

Use-case/scenario methodology – The PIE helped us confirm that the use case /scenario methodology offers a significantly more effective approach to requirements specification than the previously used informal prose approach. Users consistently rated the use case/scenario requirements specification methodology as a significant improvement over the conventional approaches based on informal prose and hierarchical decomposition that they previously used. In particular, users reported that the use of the use case/scenario approach accounted for significant improvements in many of the key attributes of a requirements specification: completeness, consistency, lack of ambiguity, verifiability and testability. In addition, the use of the new approach significantly improved the requirements when viewed as the basis for subsequent design and testing activities.

Notwithstanding these improvements, it should be noted that effort had to be invested in adapting the methodology. In particular, the UML does not prescribe how scenarios are to be specified textually, even though this is the “bread and butter” of requirements specification. We developed a structured, table-driven approach, which was then captured in our database conceptual schema and derived document formats [see figures 1 and 2].

Use case diagrams lie at the center of most expositions of the use case approach. Although these diagrams are useful, they do not play the pivotal role envisioned by the UML [7,8]. Moreover, the UML standard does not include context diagrams [4] or architecture diagrams [7,8,9] as part of the use case approach. We found that this was lacking and added them to our approach.

Database-centric vs. document centric – The approach espoused by our PIE involved a substantial paradigm shift from writing requirements documents to populating a database with requirements information and subsequently generating the needed documents automatically. The EIMS tool, like more prosaic information management systems, provides the user with forms that guide him in entering data. The data in our case consists of requirements information such as basic entities [i.e., actors, specification entities, and interfaces], behavioural requirements [i.e., functional areas, sub-functional areas, use cases, scenarios and scenario steps] non-functional requirements [non-functional areas and requirements] and related diagrams [entered via Rational Rose or Rhapsody and linked to the database]. Users appreciated the new organisational structuring principles and guidance that such a tool provides. In addition, this allowed for accessing and sharing information in ways that document-oriented approaches are not able to support. In particular, users attributed to the new tool significant improvements in individual productivity, team collaboration, the use of less experienced staff, accessibility of requirements information, the ability to inter-relate “pieces” of requirements information and assign attribute values to requirements attributes.

On the other hand, a number of problems were identified. As the tree of functional areas, sub-areas, use cases, scenarios and scenario steps grew [see figure 1], the user had some trouble visualizing and navigating the big picture. Many users requested a visual “tree” interface that would allow them to more easily navigate and manipulate the requirements tree [much like the interfaces found on modern mail and browser software applications]. This is one of the features to be provided in the next release of EIMS. Although most of the actual requirements information is deposited in the project database, this information is complemented by use-case diagrams and sequence diagrams entered using Rational Rose and connected to the database. This result is some duplication of data. For example, the user re-enters the entity names appearing in a scenario in the related sequence diagram. Similarly, use case names appear both in the database as well as in the use-case diagrams. What is needed is a more seamless interface between these two tools. The EIMS environment is currently project oriented. The unit of work is a single development project. However, in real life projects are inter-related and are often organized in product families. There have been requests to provide support for “connecting” projects, so that requirements from one project may be re-used or even shared.

Tool and methodology interaction – Most software engineering improvements involve making changes to the organization’s software development methods, tools and/or human resources. Our PIE involved making changes to all three. In such cases, care must be taken that the changes to these three elements are compatible and suitably coordinated. First and foremost, does the selected method address the improvement objectives? Second, are the new method and tool set coordinated? On the one hand, does the selected tool set effectively support the designated method? In too many cases and to the detriment of the projects involved, automated tools are adopted without investing enough thought and planning in what method the tool set is actually automating. On the other hand, has the method to be used been tailored to the capabilities and limitations of the tool set? Finally, has the staff been trained in the use of both the selected method and the supporting tool set?

To a large extent, the success of our PIE is due the efforts we made in the coordination of methods, tools and people. The method was carefully selected and defined. The tool set was tailored to the method, and the method was adapted to the limitations of the tool set. Finally, training and support was provided to baseline project staff, to ensure proper use of both the method and its supporting tool set.

Assessing process improvement – In the case of our PIE, process improvement involved an SEPG-like organization that provided improved approaches, methods, tools, training, etc. to development projects. This is a type of internal customer-supplier relationship and is fairly typical of process improvement in medium to large size development organizations. We therefore proposed using customer satisfaction measurement approaches to measuring this improvement service [14]. This involved developing a questionnaire based on the detailed objectives of the PIE, filling out the questionnaire in the course of structured interviews of key "customers" of the PIE, analyzing the results and summarizing and presenting them. This approach provides a cost-effective means of assessing process improvement. Moreover, it avoids the confounding effects of the many project parameters that cannot be controlled in the course of conducting a process improvement experiment.

Corporate Adoption - As a result of the successes of the PIE, ECI Telecom has decided to adopt this approach as a corporate recommendation. Moreover, ECI Telecom is sponsoring an internal project to port the EIMS tool set to a new development platform based on the SQL Server DBMS. This new version, EIMS-2, will also incorporate a number of new features whose need has become apparent in the course of the PIE. A number of ECI's newer development projects will be adopting EIMS-2 as their requirements engineering environment.

References

[1] Cohen, B., Harwood, W.T., Jackson, M.I., *The Specification of Complex Systems*, Addison-Wesley, 1986.

[2] Wing, J. M., A Specifier's Introduction to Formal Methods, *IEEE Computer*, September 1990.

[3] Harel, D., Politi, Michal, *Modeling Reactive Systems with Statecharts*, McGraw Hill, 1998.

[4] Yourdan, E., *Modern Structured Analysis*, Prentice-Hall International, 1989.

[5] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., *Object-Oriented Software Engineering - A Use Case Driven Approach*, Addison-Wesley, 1992.

[6] Schneider, G., Winters, J., *Applying Use Cases – A Practical Guide*, Addison-Wesley, 1998.

[7] Booch, G., Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide*, Addison Wesley, 1999.

[8] Rumbaugh, J., Jacobson, I., Booch, G., *The Unified Modeling Language Reference Manual*, Addison Wesley, 1999.

[9] Douglass, B., P., *Real-Time UML – Developing Efficient Objects for Embedded Systems*, Addison Wesley, 1998.

[10] IEEE Std. 830-1983, *IEEE Recommended Practice for Software Requirements Specification, IEEE*, 1993.

[11] Davis, A.M., *Software Requirements – Objects, Functions & States*, Prentice Hall, 1993.

[12] Koenig, S., Requirements Management Environment Selection Report, ECI Telecom, Nov. 1999.

[13] Pulford, K., Kuntzmann-Combelles, A., Shirlaw, S., *A Quantitative Approach to Software Management,* Addison-Wesley, 1996.

[14] Moller, K., Paulish, D.J., *Software Metrics – A practitioner's guide to improved product development,* IEEE Computer Society Press, Chapman & Hall Computing, 1993.

Acknowledgements

Like development projects, the success of a PIE is dependent on the dedication and contribution of many people. We would like, in particular, to acknowledge the contributions of Benny Baraz, Zvi Sivan, Boaz Sterenheim, Ido Eshed, Nir Genani and the management and development staff of the PIE's baseline projects.

Author CV

Shai Koenig has, since 1987, worked for Tadiran Telecommunications [which merged with ECI Telecom in March of 1999]. Until the merger, Shai managed Tadiran Telecommunications' corporate Software Engineering Quality and Processes Group, which was charged with establishing and improving corporate level software development processes, methods and tools. Previous positions in Tadiran Telecommunications included managing a divisional Quality & Engineering Services Department, and prior to that a divisional Software Quality & Engineering Services Department. He was appointed as a corporate Senior Scientist in 1996. Prior to joining Tadiran Telecommunications, Shai served as the Chief Software Engineer of the Communications, Command and Control Plant of Tadiran's Systems Division. Shai has twenty years of experience in software development organizations and has specialized in software processes, methods and tools.

Shai received a B.A. degree in Mathematics from Yeshiva University, an M.S. degree in Mathematics and a Ph.D. degree in Computer Science from New York University's Courant Institute of Mathematical Sciences. Shai was a member of the Computer Science Department of Rutgers University and the Mathematics Department of Baruch College of the City University of New York and has also taught at Tel-Aviv University and the Open University of Israel.

Currently, Shai is working for Comverse Network Systems, teaches in the Computer Science graduate program of the Open University of Israel and continues to provide consulting services to ECI Telecom. Shai may be contacted at ***shai_koenig@icomverse.com***

Izik Zelovich has been working for Tadiran Telecommunications [which merged with ECI Telecom in March of 1999] since 1991. Until the merger, he worked in Tadiran Telecommunications' corporate Software Engineering Quality and Processes Group, which was charged with establishing and improving corporate level software development processes, methods and tools. Previous positions in Tadiran Telecommunications included managing the Software Quality Assurance function and providing Engineering Services in the Public Switching division. Prior to joining Tadiran Telecommunications, Izik had been working in software development for a variety of companies since 1975 both on military and civilian systems. Izik has specialized in recent years in the areas of software processes, methods and tools. Previous to this, Izik taught Statistics at Tel Aviv University and worked as a statistician at the Israel Standards Institute (ISI).

Izik received both a B.A. degree and M.A. degree in Statistics from Tel-Aviv University.

Currently Izik is working for ECI Telecom in the company's Software Technology Department. He may be contacted at ***zelovich@ecitele.com***

Company Description

ECI Telecom Ltd. is a publicly owned corporation whose stocks are traded in the U.S. NASDAQ stock market. ECI Telecom is involved in the specification, development, production, sales and service of a wide range of products and systems for the telecommunications market, including transport network systems, access systems, private exchanges, computer telephony integration, data communications, and wireless access systems. ECI Telecom is Israel's largest supplier of telecommunications products, selling to over 600 customers in over 145 countries, with sales of 1.2 billion dollars per year, over 90% of which is for export. The company consists of product-oriented SBU’s [strategic business units] and market-oriented RBU’s [regional business units]. In total, the company employs over 5000 workers, of which over 1000 are working as systems and software engineers.

Dynamic CMM for Small Organisations - Implementation Aspects

Terttu Orci
Umeå University/Stockholm University
Astrid Laryd
Umeå University

Introduction

Software process improvement (SPI) is a current topic in software engineering community. Most organisations, in order to maintain their competitive edge, plan or are in progress to apply SPI with the desired results of predictability in the software development considering time, cost, and quality. SPI effort can take many forms, e.g. it can be based on a road map like Software CMM [13], SPICE, Bootstrap, ISO9000, or it can focus on improving specific working practices in an organisation. There are models for implementation, e.g. IDEAL [12], guidelines for and experiences of implementation [3],[15], and other related information, e.g. [2],[5],[6],[14].

It is commonly agreed that a software development organisation with notably symptoms of chaos caused by immaturity should start a software process improvement program. However, when a chaos is already a fact, it is difficult to find time and personnel resources for an improvement program. It would be wiser to start an improvement program before the chaos is a reality. In fact, the wisest thing to do would be to start an improvement program when starting the operation of the organisation. Obviously, we would in that case not think about an improvement, but see it as a sound business and engineering strategy.

The next best thing is to start an improvement program while an organisation is a very small one, with a few software engineers, and with a simple management. In the start-up phase, with a few software engineers, with one or a few versions of one product or only one project running at a time, the communication is simple, configuration management is relatively easy, everybody knows what the colleague engineer is doing, neither the development work nor the management offers bigger problems.

The existing models, e.g. CMM, are not, however, directly applicable for small organisations for various reasons. CMM, for example, proposes more than 25 organisational roles, with various tasks and responsibilities. In a small organisation, there is not enough people to fill the roles proposed, neither is there any need of many of those roles. Further, the models are usually described on a huge number of text pages, being cumbersome and time consuming to get oversight, digest, and apply. Therefore, the models need to be scaled down to the needs and possibilities of small organisations. A usual restriction in a small organisation is that there is not enough resources for appointing external competence for a long term SPI, but local competence is the only realistic possibility. Therefore, the models should include guidelines for internal assessment and application of the model. Further, it is essential that such a model is applicable continuously in the course of time, while the organisation grows, otherwise it will cease to be useful after a while.

We have developed a model called Dynamic CMM for Small Organisations. The model has been briefly presented in 114] and in detail in [10]. In this paper, the model is briefly presented together with some implementation guidelines.

The model is primarily intended to be used by an organisation's internal staff, with some experience of SPI and by the management. The basic idea is that the model users can easily get an overview of the model, of the magnitude of the effort required and of the roles, responsibilities, tasks, and documents. With those intentions in mind, the original CMM has been interpreted, reduced, reorganised, and partially represented in a graphical notation.

The Dynamic CMM for Small Organisations

A small organisation, according to the classification of EC, has 1-50 employees. That range represents a wide variety of organisations. The employees adding to the count are assumed to be either development staff, management or marketing and sales, excluding administrative services and human resources staff. We believe that special conditions apply to an organisation with one or two employees, because it is easy to be continuously informed of what the colleague engineer is doing. Such organisations are here called eXtra eXtra Small organisations (XXS). As soon as the organisation grows to include three persons, the loss of oversight and easy communication is at risk. Yet, an organisation with three employees is very much smaller and easier to manage than an organisation with nearly 50 employees with several products, several versions of each product or customers at a time. It seems adequate to further classify the organisations to eXtra Small (XS) with 3-15 employees and Small (S) with 16-50 employees. The models are intended to be applicable for organisations developing software products to open market, as well as for organisations developing customer specific solutions. In case there is a difference in roles or otherwise between these organisations, it is indicated in the text.

The Role Models

We have studied the roles in CMM, their interdependencies, responsibilities and activities. There is only one role, Software Quality Assurance group (SQA), that should not involve any person from the management, neither should it include persons involved in the software development. The reasons are obvious: quality assurance without impartiality does not fulfil its purpose.

The roles may be one of three types. First, a role may be important enough to be included as a role, assigned to a person with associated responsibilities, e.g. Senior Management (SM) as every organisation must have such a role irregarded the size. Second, a role may be important in terms of activities and responsibilities, but without need for a formal role. In such a case, the activities and responsibilities can be carried out by a person having some other role, with the consequence that the responsibilities and activities of that person will be more extensive than implied by his formal role. An example of such a role is Documentation Support Group (DSG) which is unrealistic in a one person company, but still there is some documentation to do, assumed to be taken over by software engineers (SE). Third, some roles are irrelevant, e.g. an organisation not applying subcontracting does not need Software Subcontract Management (SSM).

XXS Organisations

An XXS organisation is assumed to have one or two employees and one version of the first product is under way. Alternatively, the first customer project has been initiated. This is the starting scenario for many software development organisations. If there is only one employee, the person is both the manager in the role Senior Manager (SM) and a Software Engineer (SE). If there are two employees, the one is both SM and SE, the other only SE. Still, two persons have a good chance to get insight into each others work, and we see no reason to have formally assigned project management. By that, we do not claim that project management is abandoned, project planning and tracking are important activities. The implication is only that project management activities can be carried out informally by SM. SoftWare Manager (SWM) is responsible for the software development environment, but also for the operative software and hardware in the organisation. In companies developing customer specific solutions, SWM becomes the expert on computers and adjustments at the customer site. Still, a person possessing one of those roles may very well work also in another role.

We make the assumption that subcontracting is not realistic in a two-person organisation, implying that the Key Process Area (KPA) Software Subcontracting Management KPA is irrelevant. System Test Group (STG) is supposed to be responsible for system tests, while SE has the responsibility of tests during the development. Validation and verification is of primary importance to obtain a quality product. Still, an internal STG role is hardly realistic in a two person company. For organisations with customer specific development, the customer is the most adequate group for acceptance tests. In product development organisations in the beginning of the business, an external test service can be used.

SQA should not be shared with persons involved in development activities or in the management. In a two person organisation, it is not realistic to appoint an internal SQA. In an organisation with specific customers, Customer SQA (CSQA) can be used to conduct SQA activities, if SQA is available at the customer site. In product development organisations, an external SQA service can be bought.

System Engineering Group (SG) is responsible for the systems with both software and hardware. The roles in an XXS organisation are presented in Figure 1.

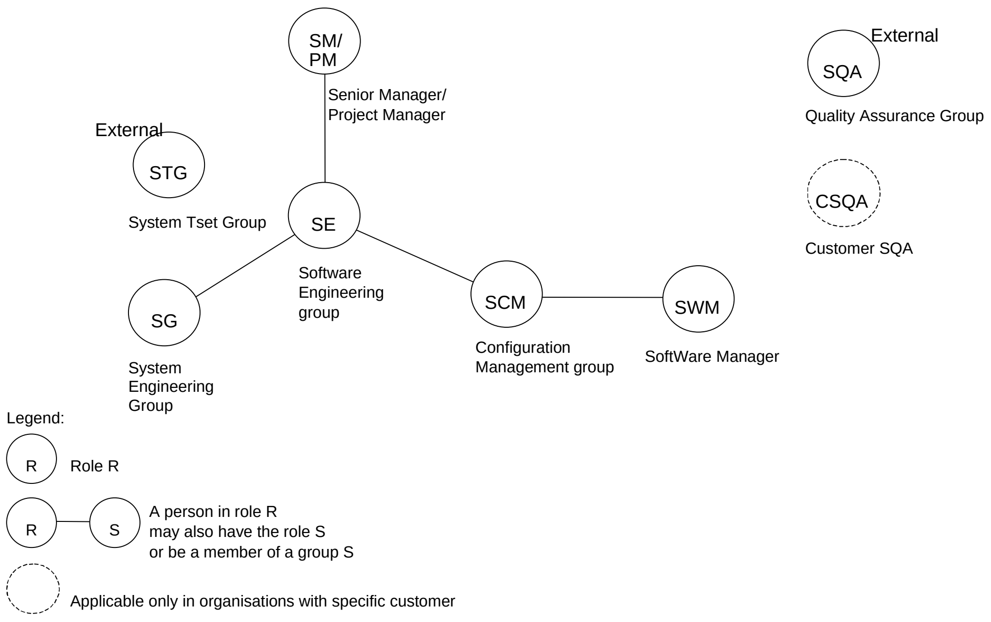


Fig. ORCI-LAR.1 The Roles in an XXS Organisation

In the model, a circle denotes a role. A dotted line around a circle means that the role is relevant only in organisation developing customer specific solutions. The links between the roles have the semantics of "shared-by", which is a binary relation, more exactly an equivalence relation. Therefore, transitivity applies. For example, the activities and responsibilities of SE may be shared by SG. From Figure 1 it is obvious that model is applicable in an organisation with only one person, who possesses the roles SM/PM, SE, SG, SCM, and SWM, and buys external services for STG and SQA, or uses CSQA, if applicable.

XS Organisations

An XS organisation has 3-15 employees, and a few products/product versions or projects. If the organisation is developing products for open market, the time of entering XS model is probably the time of the first release, i.e. the time when the hard work with successive releases and change management begins. Moreover, the life after the first product is of strategic importance. In order to keep the first product successful and on the market, the organisation needs to grow. Most probably, new products are on the way, also implying need to recruit more staff. Going from XXS to XS, new roles are needed. Correct project management requires Project Manager (PM) role. Most probably, marketing and sales (MS) is a role with increasing importance at the time an organisation grows out from XXS. System testing (STG) should become an internal role as the number of products/product versions and projects grows.

Figure 2 presents the model for XS organisations. The shading of a circle indicates that the role is an additional one compared to the XXS organisation. For example, when growing out of XXS, PM and MS are roles to be added. The frames depict projects. The roles within a frame denote roles, which may be specific for each project, while the roles outside the frame are roles having responsibility in the whole organisation. For example, there is SCM role in each project. SWM is general for the organisation, but the person in SWM role may belong to SCM in one or several projects. SM and MS are general for the organisation, possibly one and the same person. SM can also work as SE.

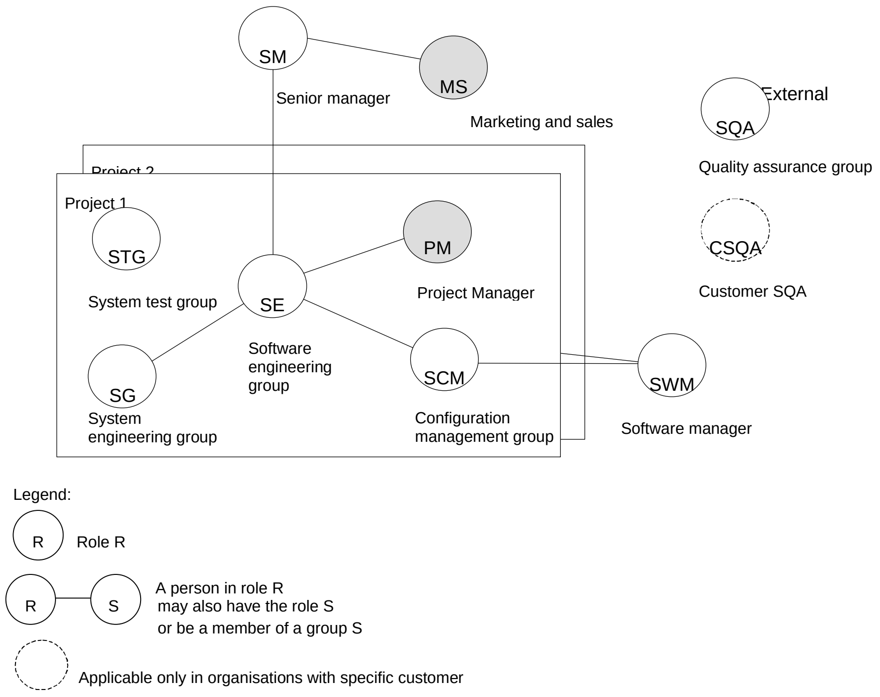


Fig. ORCI-LAR2. The Roles in an XS Organisation.

S Organisations

In an S organisation, the number of employees is 16-50, and there are several products/versions or projects running in parallel. In particular for organisations with product development, documentation support is of great importance implying introduction of Documentation Support Group (DSG). Also, software subcontracting is relevant. This implies the introduction of the Key Process Area (KPA) Software Subcontract Management with the role Software Subcontract Manager (SSM). In some organisations, SM may not be involved in MS activities, but in some organisations, sharing SM and MS by one persons is considered important. Therefore, we leave this possibility open by retaining the link between MS and SM. The links imply possibility, not a mandatory sharing of responsibilities. MS is related to DSG, and the same person can share DSG and MS roles. Further, until subcontracting is widely practised, the person in role SM may also have the role SSM. Quality assurance in this size of the organisation motivates an independent internal SQA, which is also important for obtaining continuity in the software process improvement activities. SCM is a more requiring task in an organisation with several projects/products/product versions, and therefore it is assumed that the person in SCM role does not also work as SE. The model for an S organisation is presented in Figure 3.

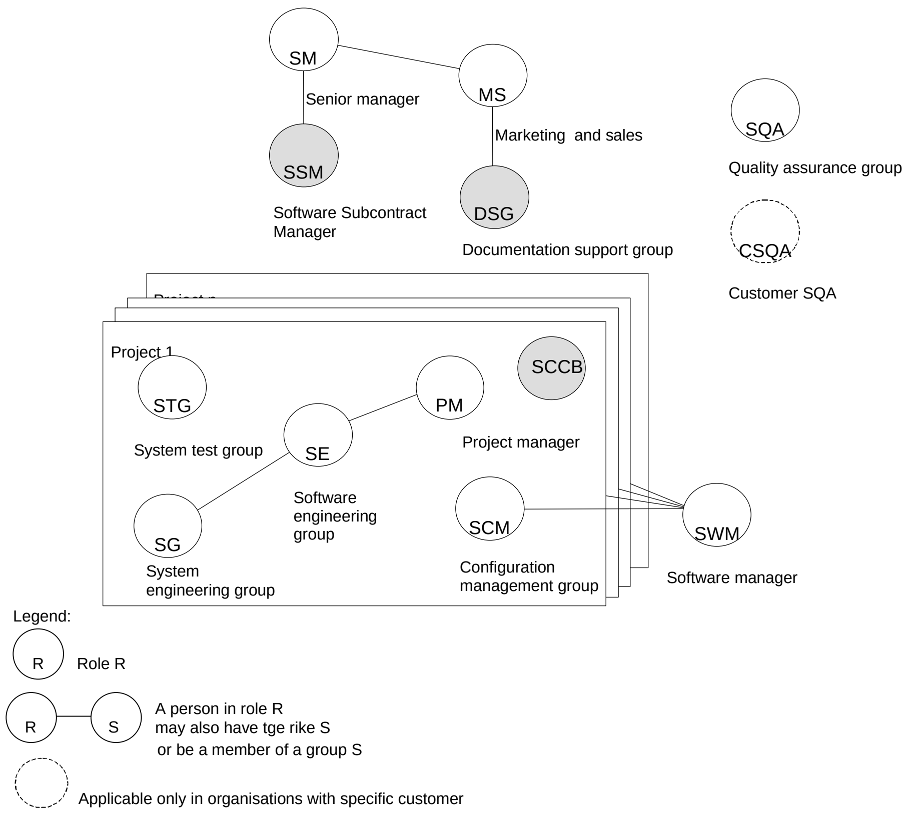

SM
Senior manager
SSM
Software Subcontract Manager
MS
Marketing and sales
DSG
Documentation support group
SQA
Quality assurance group
CSQA
Customer SQA
Project n
Project 1
STG
System test group
SE
Software engineering group
PM
Project manager
SCCB
SG
System engineering group
SCM
Configuration management group
SWM
Software manager
Legend:
R
Role R
R
S
A person in role R may also have tge rike S or be a member of a group S
Applicable only in organisations with specific customer

Fig. ORCI-LAR3. The Roles in an S Organisation

Responsibilities and Activities

For each role, there are responsibilities and activities. Here, only the responsibilities and activities for one role, SWM, are described as an example of the representation, in Table 1.

Table 1. The Responsibilities and Activities of SWM

Type	Object of concern
Is responsible for	- the development environment HW and SW
Reviews	-statement of work -project plan -resource estimates -quality assurance plan -configuration management plan -the progress against the project plan -specification for subcontract -subcontract

Documents

CMM requires a large number of documents, required as the basis for activities or as input or output to/from activities. The phrase "according to a documented procedure" appears frequently in CMM text. The documents can be classified as follows:

- policy documents - one for each KPA
- documented procedures, e.g. requirements management, project planning, estimation
- plans, e.g. configuration management plan, project plan, quality assurance plan
- status reports, e.g. quality assurance report

In CMM, the documents are described in text, but it is hard to get oversight of the number and the types of documents. In our model, documents are classified by type along the above document types. The documents are not described here for space reasons, but can be found in [10].

Activity Diagrams

One of the intentions with the model is to get an easy overview. Graphics support both overview and structure and therefore, the activities are represented graphically, in a simplified form. A graphical overview of only the KPA Software project Planning is given in Figure 4. The activities take allocated requirements as input, and the output consists of statement of work, estimates, project administrative data, and project plan. The activities are reviewed periodically with SM, and periodically as well as event-driven with PM. SQA reviews the activities and work products of the KPA, indicated by SQA outside the KPA frame. SM is responsible for a written organisational policy, and a set of documented procedures in project planning. The PM is responsible for measurement data concerning the status of project planning activities. An overview of the SPP KPA is given in Figure 4 and the activities in Figure 5.

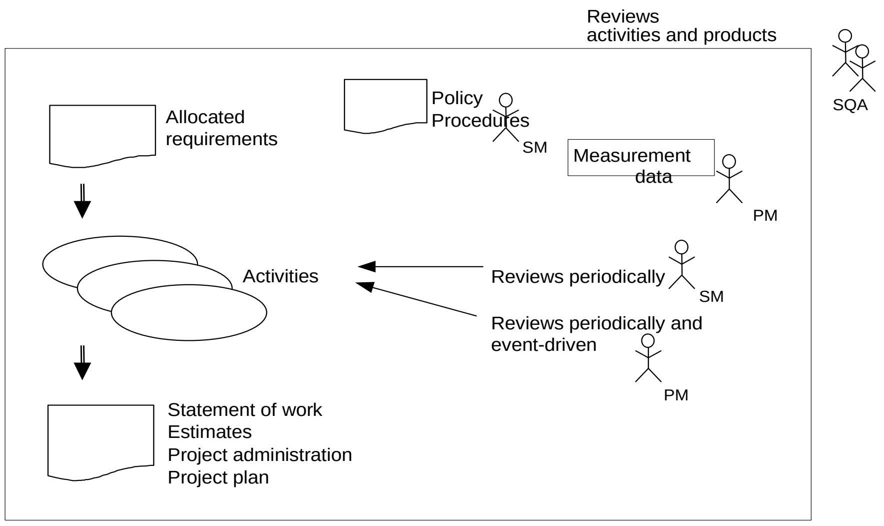


Fig. ORCI-LAR4. Software Project Planning - overview

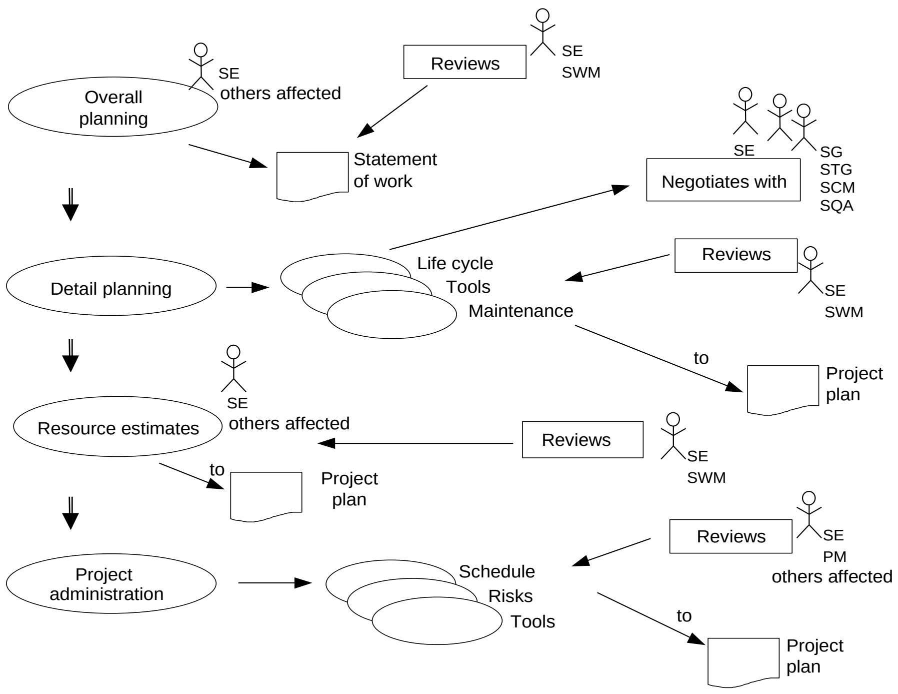


Fig. ORCI-LAR5. Software Project Planning - the Activities

Implementation of Dynamic CMM

In this section, guidelines for implementation of Dynamic CMM in small organisations will be presented. We do not argue that the guidelines are the best, or even appropriate for all situations. To do that, the guidelines must be used in a number of real situations, followed by systematic analyses of the results, and of the opinions of the practitioners. The guidelines presented here have been developed by common sense, experience of ours and others, and should so far be regarded only as a proposal.

A software process improvement effort should be driven in a project form, with project plan, resource estimates, budget, and schedule. It is also important that the project management of the improvement process is conducted in the way proposed in Software Process Planning KPA at Level 2 for development projects, to serve as a good picture of what is expected from management issues in the development projects. In order to maintain the focus and to prevent misunderstandings in the following discussion about projects, processes, and roles, a distinction must be made between software project improvement and development. For both improvement and development, we can distinguish projects and processes. The terms I-Project, D-Project, I-Process, and I-Project are used to make a distinction. A long term improvement effort is called an I-Process. An I-Process involves some particular steps to be taken initially, and a number of iterations of smaller improvement efforts, realised in project form, in I-Projects. Each of the I-Projects includes planning, doing and analysing the results. The doing coincides with design and use of a D-Process, e.g. one particular KPA like Requirement Management. The use of the newly designed D-Process is in a pilot D-Project. Also for the roles needed in the improvwment work, the distinction is useful. The roles prescribed by CMM are roles in software development organisation, in the development work, and are called D-Roles. The roles needed for SPI work are called I-roles, e.g. SEPG. The distinction between improvement and development is illustrated in Figure 6.

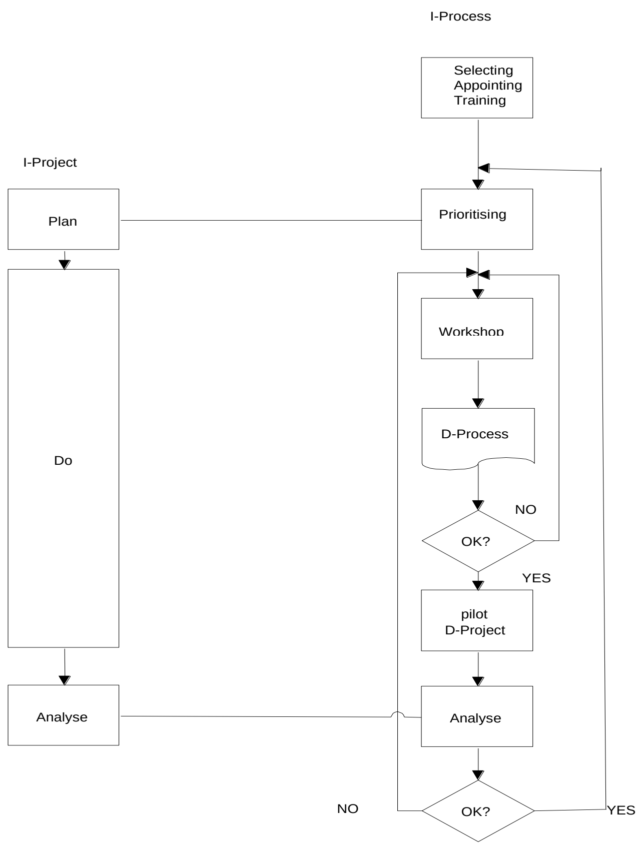


Fig. ORCI-LAR6. An I-Process including an I-Project

The activities of an I-Process are described in detail below.

Selecting - Appointing - Training

Selecting

The basic parameters for selecting an appropriate model are the number of employees and the number of products/product versions or projects. The number of employees includes development, management, and marketing and sales staff, but excludes administrative and human resources staff. The number of products and product versions is applicable for product development organisations, while the number of projects is applicable for organisations developing solutions to specific customers. Table 2 gives the basics for the decision of an appropriate entry point to the model.

Table 2. The entry points to Dynamic CMM

	1-2 empl	3-15 empl	>15 empl
1 product/version or project	XXS	XS	S
2-5 products/versions or projects	-	XS	S
>6 products/versions or projects	-	S	S

The alternatives marked with "-" are unusual if not unrealistic. For example, an organisation with 2-5 products/product versions, with 1-2 employees might have a theoretical chance to stay on the market with a large number of subcontractors, but it is more a rule than an exception that the company growth implies growth in the number of employees. If the organisation characteristics are outside the limits of the table, it is assumed that the original CMM can be used.

Appointing

The I-Roles for SPI-work are *steering group*, *SEPG*, *working groups*, *process users*. The *steering group* decides which D-Processes are to be improved during the closest time period, selects working groups, and appoints SEPG. In a small organisation, the steering group may very well consist of the senior manager only. The *SEPG* (Software Engineering Process Group) is not required until Level 3 in CMM. However, there should be a change agent or an opinion leader, or group of those, with an interest in SPI and knowledge of both SPI and software development. That role is here called SEPG. The tasks of SEPG are to collaborate with the working groups in SPI work, to co-ordinate the details in realisation of the vision, to train the working groups and process users, to support and motivate the use of the process. The *working groups* use their knowledge and experience to do the main work in defining and evaluating the new processes, and train the process users. In a small organisation, there might not be any people being only *process users*, but everybody in the development teams might be involved in working groups. There might be one or several working groups, depending on the situation at hand.

Training

The training should concern the underlying ideas and motivations for SPI work in general, and the Dynamic CMM in particular, including the roles, responsibilities, tasks, and documents. The role models, the list of responsibilities and activities, and documents, are intended to serve the purpose. The activity diagrams, both the overviews and the detailed ones, give a picture of the KPAs, corresponding to D-Processes.

Prioritising

PRIO stand for prioritising, implying selection of one or more KPAs for the improvement work. The order of KPAs is not fixed by the original intentions of CMM. However, assuming that there are no defined and documented processes, but chaos and reactive way of work, it would not make much good to introduce SQA as the first priority. At least one other KPAs should be in place before SQA is meaningful. Project management is important as it has consequences for the delivery time and budget. However, if the entry point is XXS, PM is informally handled. The order of the formally handled KPAs, Requirement Management (RM) and Software Configuration Management (SCM), is not generally determined. Which one is introduced first is a question of the situation at hand. For example, if configuration management is experienced as a big problem, it would be natural to start to bring some order in that. At the other hand, if the requirements are volatile, and problems are encountered by introduction of new requirements, RM should probably get first priority.

The selection of the high priority KPA is not independent of the model selected. If the model is XS or S, SPP and SPTO are maybe the most adequate KPAs to start with. For the order of RM and SCM, the same reasons should apply as in XXS organisation.

The prioritising activity is the first task in Plan activity in an I-Project. Further planning in an I-Project requires, among other things, that appropriate resources are given by the steering group, that people are appointed to the organisational roles for the KPA in case, and that a project plan for the I-Project is developed and documented.

Workshop

Workshop is a suitable form for defining D-processes. The reasons and motivations for this technique have been presented in [3]. The final result of a series of workshops is a defined and documented D-process, e.g. a process for Software Project Planning. The form and content of respective D-process should conform to the activity diagrams for KPAs. In order to check whether the defined D-Process satisfies the model requirements, we have developed checklists for each KPA as well as for each activity within a KPA. The checklists include entry conditions, roles, activities, and exit conditions. The checklist for the Software Project Planning KPA (D-Process) is presented in Tables 3-5. The checklists for SPP activities are omitted for space reasons, but can be found in [13]. A defined and documented D-process must be approved by the working group and SEPG. If an approved status cannot be directly reached, a new workshop should be arranged to solve the problems and modify the D-Process.

Entry conditions		
	A PM is designated to be responsible for the project plan	❑
	Written policy	❑
	Adequate resources and fundings are provided for the planning the software project	❑
	PM, SWM, SE, and other individuals are trained in software planning and estimates	❑
	Tools to support the activities are made available	❑
	Documented procedures for developing the project's plan	❑
	Allocated requirements	❑
	Cost data (optional)	❑
	Historical data (optional)	❑
	Project proposal	❑
	Project and customer standards	❑
	Statement of work	❑
Roles		
	Senior Manager (SM)	❑
	Project Manager (PM)	❑
	SoftWare Manager (SWM)	❑
	Software Engineering group (SE)	❑
	SQA group	❑

Table 3. Software Process Planning Checklist - Entry Conditions and Roles

Activities		
1	SE participates on the project proposal team	❑
2	SE reviews the proposed commitments	❑
3	SE reviews the overall project plan	❑
4	Commitments made to individuals and groups external to the organization are reviewed with SM	❑
5	A software life cycle with predefined stages of manageable size are identified or defined	❑
6	The project's plan is developed	❑
7	PM, SWM, and other affected groups review the project's plan	❑
8	The project's plan is documented	❑
9	Software work products that are needed to establish and maintain control of the software project are identified.	❑
10	Estimates for the size of the software work products (or changes to the size of software work products) are derived	❑
11	Estimates for the software project's effort and costs are derived	❑
12	Estimates for the project's critical computer resources are derived.	❑
13	The project's software schedule is derived	❑
14	The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented	❑
15	Plans for the project's software engineering facilities and support tools are prepared.	❑
16	Software planning data are recorded	❑
17	Measurement data are made and used to determine the status of the software planning activities	❑
18	The activities are reviewed with SM on a periodic basis	❑
19	The activities are reviewed with PM on both a periodic and event-driven basis	❑
20	SQA reviews and audits the activities and work products and reports the result	❑

Table 4. The Checklist for Software Project Planning - the Activities

Exit conditions		
	Action items resulting from reviews with PM and SM are documented and reviewed	❑
	Assumptions made in deriving the estimates for the project's effort and costs are documented and reviewed	❑
	Assumptions made in deriving the estimates for the project's schedule	❑
	Distribution of effort, staffing and cost estimates are prepared	❑
	Estimates for critical computer resources size (or changes in size) of software work products project's effort and costs capacity requirements for software engineering facilities and support tools	 ❑ ❑ ❑ ❑
	Mesurements	❑
	Plans for involvement of SE in activities of other groups	❑
	Plans for other groups involved in SE activities	❑
	Plans for software engineering facilities and support tools	❑
	Project's software schedule	❑
	Software life cycle	❑
	Software project commitments	❑
	Software risks	❑
	Work products	❑
	Summary report from each review with SM, PM	❑
	Time phasing activities	❑

Table 5. The Checklist for Software Project Planning - the Exit Conditions

Pilot D-Project

Pilot D-Project is simply an application of the defined and documented D-Process in a real development situation, in a D-Project. The pilot intends to result in opinions of what worked well and what did not work well.

Analyse

The opinions of the process users are analysed after the pilot D-Project. If the D-Process cannot reach the approved status, a workshop should be arranged to solve the problems and to modify the D-Process accordingly. The intention is that the D-Process is used in every D-Project after its definition. Until the D-Process has reached the approved status, it should also be analysed and modified after each application.

Concluding Remarks

In this paper, we have briefly presented the model called Dynamic CMM for Small Organisations. The approach is not unique, there is some related work on the issue.

The work presented in [1] describes the issues encountered by small businesses, organisations, and projects, if applying the existing models for SPI, e.g. CMM: documentation overload, layered management, scope of reviews overkill, limited resources, high training costs, and unrelated practices. A set of tailoring guidelines for each of the issues is presented in [1], and a tailored CMM, based on the original CMM with marked additions and deletions. The same authors present more detailed tailoring guidelines in [7] for different issues, e.g. documentation tailoring, review tailoring. It is important to emphasise the scalability issues of Software CMM for small organisations. We believe, however, that not only should the model be downscaled, but also the representation should be restructured, e.g. graphics, lists, for simplicity and easy overview. In [9], SPI work in small organisations has been presented, and critical success factors given, they being a flexible, tailored assessment and improvement approach, a functioning network of small enterprises and their environment, external technical help by mentors for change, and external financial support coupled to some conditions for performance. Our model is intended to support the first factor. Although external technical help may be needed to take the first steps in the SPI work, we believe that only organisations with an internal strong competence in SPI have the chance to conduct long-term SPI work. External services should only be needed for assessments to obtain impartiality.

The current research concerns validation of the model in real cases, one being in progress and two more under negotiations. The real cases involve not only application of the model in SPI work in the organisations in case, but empirical evaluation of the model and of the underlying ideas.

References

[1] Brodman J.G., Johnson D.L. The LOGOS Tailored CMM for Small Businesses, Small Organizations, and Small Projects. LOGOS International, Inc.1995.

[2] Byrnes P, Phillips M. Software Capability Evaluation, Version 3.0, Method Description, Technical Report CMU/SEI-96-TR-002, ESC-TR-96-002, 1996.

[3] Caputo K. Implementation Guide - Choreographing Software Process Improvement. Addison Wesley, 1998.

[4] Curtis, B.,Hefley, W.E., Miller, S. People Capability Maturity Model. Software Engineering Institute, Carnegie Mellon University, 1995.

[5] Dunaway D.K. et al. Why Do Organizations Have Assessments? Do They Pay Off?, Technical Report, CMU/SEI-99-TR-012, ESC-TR-99-012, 1999.

[6] Hayes W., Zubrow D. Moving On Up: Data and Experience Doing CMM-Based Process Improvement, Technical Report CMU/SEI-95-TR-008, ESC-TR-95-008, 1995.

[7] Johnson D.L., Brodman J.G. Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects. LOGOS International Inc, USA.

[8] Kautz K. Software Process Improvement In Very Small Enterprises: Does It Pay Off?, Software Process - Improvement and Practive 4, 209-226, 1998.

[9] Kautz K. Even in Very Small Software Enterprises Metrics Can Make Sense!, in the Proceedings of IRIS 21, 1998.

[10] Orci, T., Laryd, A. CMM for Small Organizations, Level 2. Umeå University, UMINF-00.20, 2000.

[11] Laryd, A., Orci, T.: Dynamic CMM for Small Organizations. In Proceedings of ASSE2000, Argentina, 2000.

[12] McFeeley B. IDEAL - A User's Guide for Software process Improvement. Handbook, CMU/SEI-96-HB-001, 1996.

[13] Paulk, M.C. et al. The Capability Maturity Model - Guidelines for Improving the Software Process, Addison-Wesley, 1995.

[14] Stelzer D, Mellis W. Success Factors of Organizational Change in Software Process Improvement, Software Process - Improvement and Practice 4, 227-250, 1998.

[15] Zahran, S. Software Process Improvement. Practical Guidelines for Business Success. Addison-Wesley, 1997.

CV Terttu Orci

Terttu Orci received her PhD at the Royal Institute of Technology (KTH), Stockholm, and Associate Processor at Stockholm University. She works at lecturer, researcher, and supervisor at the department of Computer and Systems Sciences, Stockholm University/KTH. She is in charge of the competency track in Software engineering in the education programmes at both universities, with special focus on software process improvement and software metrics. Although the current research interest is in software engineering, the prior research interests include databases, temporal aspects of database modeling, and automated theorem proving and AI. She is often appointed by the EC as external expert in evaluations of project proposals and in projects. She also works as consultant in industry in various cooperation project between academia and industry. Currently she focuses on initiating SweSMA, Swedish Software Metrics Association, to be included in FESMA.

CV Astrid Laryd

Astrid Laryd received her M.A. at the University of Lund, Sweden. She has worked in the Swedish industry both as an employee and as a consultant. Examples of work are design and implementation of assemblers for minicomputers, construction and implementation of mathematical library functions for compilers, systems analysis and programming for processing documents using optical character recognition technique, design and implementation of device handlers for graphic equipment and for data panels and menus. The last ten years her work has been focused on safety in software intensive system (risk and safety assessments in nuclear power plants) and on software process improvement. She has also had a commission as a reviewer in a EU project (ESPRIT 22187); “Safety and Risk Evaluation using Bayesian Nets”, addressing the safety justification process for software-intensive system. Currently she is engaged at Umeå University in a research project: “Quality Management for Small Enterprises (QMSE).

Session 2 - SPI and People/Skills

Session Chair: Bernd Hindel, 3Soft

The Importance of NOT Learning from Experience

Magne Jørgensen
Industrial Systems Development, Department of Informatics, University of Oslo, Norway

Dag Sjøberg
Industrial Systems Development, Department of Informatics, University of Oslo, Norway

1 Introduction

"What we learn from history is that people don't learn from history." (George Bernard Shaw)

In this paper we argue that learning correctly from software development experience can be a difficult task and that, frequently, there is not enough information available to draw valid conclusions. In our opinion, a focus on how to learn from experience should include a focus on how to avoid incorrect learning from experience. This includes a focus on when we should **not** learn from experience.

We have tried to transfer general findings from human judgement studies and other sources to a software context and, in particular, to an experience database context. We believe that it is especially important to avoid incorrect reasoning from experience when establishing experience databases to spread experience within an organisation. The lack of focus on how to avoid incorrect reasoning may be a major reason for, as far as we have observed, the lack of success of implementation of software development experience databases. If we cannot trust the contents of an experience database, or the contents do not lead to better judgement processes, then experience databases are not very useful.

The remainder of this paper is organised as follows. Section 2 gives an example illustrating problems we face when attempting to learn from experience. Section 3 discusses typical learning from experience fallacies and techniques to avoid them. Section 4 concludes and describes further work.

2 An example

A project leader summarises lessons learned from previous projects that he has lead. The project leader remembers that most (approx. 80%) of his projects with time overruns had an incomplete requirement specification (at the time when the design phase was started). The project leader, consequently, draws the conclusion that an incomplete requirement specification means a high risk of late delivery.

This conclusion may look rather obvious, but is in many ways a typical example of high risk of incorrect learning from experience:

1) If the incomplete requirement specification is not the cause, only a correlating factor, then the observed relation may not be present in the future. In this example, a more likely reason for late delivery is not the incomplete requirement specification alone, but the combination of incomplete requirement specification, use of project models assuming fixed and complete requirements (for example, a waterfall model) and lack of proper requirement management. Looking at only one of these variables in isolation may not be meaningful. This error is the frequently made "preceding is causing"-error, see [4].
2) If the project leader strongly believes that incomplete requirement specification is a major reason for project delays, this will impact how he interprets project events. For example, the classification of requirement specifications into incomplete and complete can be a highly subjective task. In particular, this classification can be biased when it is carried out after the completion of the projects. This error is an example of "selective perception", discussed in Section 3.
3) It is not sufficient to remember the projects with incomplete requirement specifications and their proportion of "time overruns". It is just as important to remember the projects **without** incomplete requirement specifications and their proportion of "time overruns"! Assume that the project leader has carried out 18 projects distributed as follows:

	Incomplete specification	Complete specification
Time overrun	10 projects	2 projects
On time	5 projects	1 project

Table 1 – Project distribution

The project leader correctly summarises that approx. 80% (10/(10+2)) of the projects with time overruns had an incomplete requirement specification. This does, however, not mean that an incomplete specification means a higher risk of time overruns. The frequency of time overruns given a complete requirement specification (2/(2+1) = 67%) is exactly the same as the frequency of time overruns given an **incomplete** requirement specification (10/(10+5) = 67%). In other words, the completeness of the requirement specification is of no use to predict time overrun. This error is an example of the "neglect of base rates"-error [4]. We do not remember the situations where an incomplete requirement specification is not followed by a time overrun as well as the situations where an incomplete requirement was followed by time overruns, and/or do not use the available information about the frequency of the scenarios properly.

The project leader has, of course, much more information regarding the consequences of the incomplete requirement specifications in each projects, for example, the degree of rework needed and the frustration of the project members. This detailed scenario information can be useful, but can also lead to "representative thinking" [19] and other "learning from experience"-problems discussed in Section 3. Contrary to what our "intuition" tells us, more information does frequently lead to worse explanation or prediction models [19].

3 How to avoid incorrect learning from experience

This section discusses a small, but important, selection of "learning from experience"-problems and how to avoid them. The selection and discussion are far from complete and we recommend [1, 2, 4, 7, 8, 10, 15, 16, 19] to get a more complete picture.

3.1 Creation of experience

Selective perception: The same events are perceived differently by different people. Studies show that what we perceive is heavily influenced by what we expect and want to find [19, 24]. For example, a study [11] of how different football supporters perceived a football game concluded: *"It seems clear that the 'game' actually was many different games. ... It is inaccurate and misleading to say that different people have different 'attitudes' concerning the same 'thing'. For the 'thing' simply is not the same for different people whether the 'thing' is a football game, a presidential candidate, communism, or spinach*". Transferred into a context of software development experience database, this means, for example, that a software project experience database should reflect differences in perceptions. Use of multiple perspectives and an acceptance that a software project is not (opinions on) the same thing for all actors may reduce the risks of selective perception. The experience collector should ask the experience providers to consider reasons why their perception may be selective or biased.

Hindsight bias: When people know the actual outcome of a process, they tend to regard that outcome as having been fairly predictable all along – or at least more predictable than they would have judged before knowing the outcome [17]. Experience from software projects collected after their completion can be strongly impacted by hindsight bias ("I-knew-it-from-the-beginning..."). Unfortunately, it is not enough to inform people about hindsight bias and encouraging them to avoid it [9]. To reduce the size of the hindsight bias, people should be asked to consider how alternative outcomes might have occurred.

Improper learning models:
Several studies [1, 6, 14] report a disappointingly low correlation between length of experience and the quality of professional's judgements. An improper learning model is one important reason for this [1]:

- We try to confirm theories, rather than reject incorrect hypotheses.
- The fact that we are able to formulate a rule is often sufficient to believe that we have a good rule even though we have no experience indicating that the rule is valid. That is, the confidence in own knowledge increases with the ability to find rules regardless of the validation of these rules.
- In cases where we act on the experience based judgement there will be a number of additional factors that prevent us from detecting that our judgement is incorrect, e.g. self-fulfilling prophesies.
- We tend to prefer deterministic rules (IF <a> THEN <b> - type of rules) even if the relationship between variables is probabilistic (IF <a> THEN 20% probability of <b> - type of rules). If we find no deterministic rules, we tend to assume that there is no rule at all and start guessing. Applied on software projects this means that we tend to prefer rules where the same software project characteristics lead to the same project outcomes each time, see [15]. We do this in spite of the fact that a probabilistic rule where similar software projects may have several possible outcomes, each of them with a connected probability, would be more appropriate. The very limited use of probabilistic software effort estimation models in the studied organisation supports that this preference of deterministic rules is the case for maintenance work, too.

A key to improve learning from experience in a probabilistic environment is, according to [1] and [10], that we are able to detect the probabilistic nature of probabilistic tasks. However, the only way a person can detect this is to evaluate the usefulness of deterministic rules against probabilistic rules. This requires that the person must have the hypothesis that the task may be probabilistic and know how to test for this. This knowledge can, according to [1], hardly be derived from experience alone, but must be taught. Proper education and proper concepts of probability based relations may thus be necessary in order to learn from experience in a highly probabilistic environment [10].

Usefulness of experience: Much of our experience is of no value for future work:

- Change of conditions makes history invalid. Sometimes the experience itself changes the conditions so much that the experience is invalid. The change of conditions is particularly frequent in a software development context.
- Frequently, it is impossible to compare what actual happened with what had happened if the actual actions were not taken. A cause effect is, therefore, very difficult in one of a kind

development projects.

- Experience is context dependent and much of the context is hard (impossible?) to describe. One of the authors of this paper once tried to extract useful information from a number of project experience reports in a large Norwegian company, but found that the lack of context information made the use of those reports for other projects impossible [25].

It is important to reflect on the usefulness of experience and try not to be lead by our intuition of how important our experience is. Our intuition of the usefulness of our own experience is easily impacted by irrelevant factors, such as, the degree of effort, amount of frustrations or the emotions involved. We should accept that sometimes there is nothing or very little to learn from experience, even when we have worked hard and a lot of very interesting events have occurred.

3.2 Use of experience

Software development environments are complex, and, frequently, we have not the time, the competence and/or possibility to collect and process all information necessary to make 100% valid conclusions. We therefore use "heuristics" (rules of thumb) based on the available experience and information to make decisions or extract general rules at a, hopefully, sufficient level of quality. Most of the heuristics are unconscious and, for this reason, hard to analyse and change by the subjects themselves.

Availability heuristic: One frequently used heuristic is the "availability heuristic" [22], which states that we *"assess the frequency of a class or the probability of an event by the ease with which instances or occurrences can be brought to mind"*. Usually, this heuristic works quite well, common events are easier to remember than uncommon events. There are, however, situations where this heuristic leads to incorrect judgements. Some events are easier to remember because they are inherently easier to think of, because they have taken place recently or because they are highly emotional. The improper use of this heuristic may be the main reason for our finding in [13], where the software managers believed that their maintenance groups use much more effort on software corrections than the groups actually did. Software corrections, probably, receive more management attention compared with other types of maintenance work and are, therefore, overestimated.

One way to avoid the problems connected with the availability heuristic may be to keep records of the software development work and frequently update the records. Post-mortem project reviews (experience reports) based on what the project members remember at the end of the projects may be better than no experience collection. On the other hand, if no experience is recorded during the projects (for example, in a project diary) much of the information collected may be very much biased due to the availability heuristic.

Anchoring and adjustment: Anchoring and adjustment may be the most common heuristic when reasoning by analogy. This heuristic consists of two steps:

Make an initial judgement (an "anchor").

Adjust the initial judgement for differences between current situation and the "anchor" situation.

Studies [18, 22] indicate that the adjustments are, most of the time, insufficient and that the anchor has an unexpected high impact on the final judgement. This heuristic is used when experts are estimating software project effort. The estimation experts find the closest analogies (the anchor projects), for example using the "representativeness heuristic" (see explanation later in this section). Then the estimation expert adjusts for differences between the anchor projects and the project to be estimated.

As found in [12], an estimation process based on "anchoring and adjustment" (expert estimation) performs well compared with other estimation methods when the anchor projects are good predictors of the new project. This process leads, however, to lower accuracy than more statistically based estimation models when the anchor projects is quite different from the project to be estimated. The reason for this low accuracy is the inability of expert estimators to adjust sufficiently for large differences in project characteristics.

The problems people have in adjusting insufficiently from anchor values seem to be a very robust effect that is difficult to prevent. Frequently we are unaware of what we use as anchor values. The maybe only way to avoid the problems and, at the same time, keep the advantages of this heuristic is to increase awareness of the heuristic and to know when to use more formal approaches, such as statistically based prediction methods.

Interestingly, the knowledge of this human weakness (that we do not adjust sufficiently from the anchor based value) can be used to compensate for another human weakness; the tendency for planners to make too optimistic prediction. The compensation technique, used by some project leaders, is to start the estimation process with the "most pessimistic" effort estimate and then estimate "most likely" and "most optimistic" effort. The "most pessimistic" effort estimates will then be the "anchor values" and lead to lower estimates of "most likely" effort compared with an estimation process starting with the "most likely" effort estimates. Results supporting this finding are reported in [3].

Representativeness heuristic: Another common heuristic is the "representativeness" heuristic [22], i.e., a judgement process based on the similarity between A and B and the assumption that similarity with respect to some properties means similarity with respect to other properties. This heuristic is a type of analogy based reasoning. The representativeness heuristic works well in most cases, but can be a major source of biased use of experience in some cases, as reported in [23], p 98:

"*As the amount of detail in a scenario increases, its probability can only decrease steadily, but its representativeness and hence its apparent likelihood may increase. The reliance on representativeness, we believe, is a primary reason for the unwarranted appeal of detailed scenarios and the illusory sense of insight that such constructions often provide.*"

The results we describe in [14] exemplify this. Interviews with software maintainers indicated that they used the representativeness heuristic together with the "anchor and adjust" heuristic when predicting problems connected with solving a software maintenance task. The maintainers tried to remember a similar task carried out recently (the anchor) and based their predictions on the degree of problems conducting that task adjusting for differences. This heuristic did not work well for the software maintainers predicting maintenance problems. Although they had detailed knowledge about the task to be completed by themselves, a simple decision model based solely on the historical rate of major problems for large/medium and small maintenance tasks outperformed the prediction of the maintainers. Similar results are reported in, for example, [5].

The representativeness heuristic is based on (or influenced by) the assumption: "The more similar A is to B, the more likely A will behave similarly to B". This assumption may be correct most of the time and is a very useful heuristic. To avoid the misuse of this heuristic we must, however, understand the limitations. For example, consider a new software project A, which is very similar to a completed project B, which in turn had an extremely high productivity. This extremely high productivity is not only caused by the variables that make project A similar to project B, but also to other variables, such as an unusual degree of luck. Then, it is unlikely that project A will have the same high productivity as project B (regression to the mean-effect [21]). In this case, it would have been better to use less similar projects as a basis for the estimation of the productivity of project A.

3.3 General guidelines

The suggestions on how to achieve the goal of not learning from experience can be summarised as:

- Ask the experience providers to consider alternative perspectives. Stimulate critique of own judgements. (This guideline is, in our opinion, the most important and robust guideline on how to avoid biases and misuse of the heuristics.)
- Learn about the biases and the limitations of the heuristics and increase the awareness of them. Note, however, that learning about the biases and limitations of the heuristics is not always sufficient to eliminate them.
- Conduct training (and give feedback) on how to avoid the biases and the misuse of the judgement heuristics.

- Use statistical techniques and increase the statistical knowledge of the software professionals. In particular, skills on how to establish probability-based models of relationship between project characteristics are important, we believe.

4 Conclusions and further work

Learning correctly from experience is difficult. Without knowledge and reflections on the biases and limitations of the judgement heuristics:

- much of the learning will be incorrect,
- experience databases and "post mortem reviews" (experience reports) will include much incorrect information, and
- invalid conclusions about causal relationships will be drawn.

The perhaps simplest and most efficient way to reduce the negative impact of the biases and judgement heuristics is to ask the software professionals to consider alternative perspectives and to stimulate critique of own judgements. In addition, we need to increase the awareness and knowledge of software professionals and researchers.

The problems of learning from experience have been studied for at least 60 years. Surprisingly, the work on experience databases in software organisations does not refer to or incorporate the knowledge from these studies. We believe that it should.

This paper described the beginning of our effort on how to improve learning from experience through tools and processes based on knowledge about biases and judgement heuristics. We have already started collecting data in collaboration with a major Telecom company.

Acknowledgements

We gratefully acknowledge the collaboration with our industrial partner, Ericsson. The research project is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry).

References

[1] B. Brehmer, "In one word: Not from experience," *Acta Psychologica*, Vol. 45, 1980.
[2] C. F. Camerer and E. J. Johnson, "The process-performance paradox in expert judgment," in *Toward a general theory of expertise*, K. A. Ericsson and J. Smith, Eds. Cambridge: Cambridge University Press, 1991, 195-217.
[3] T. Conolly and D. Dean, "Decomposed versus holistic estimates of effort required for software writing tasks," *Management Science*, Vol. 43, 1997, pp. 1029-1045.
[4] R. M. Dawes, *Rational choice in an uncertain world*: Harcourt Brace & Company, 1988.
[5] R. M. Dawes and B. Corrigan, "Linear models in decision making," *Psychological Bulletin*, Vol. 81, 1974, pp. 95-106.
[6] S. M. Doane, J. W. Pellegrino, and R. L. Klatzky, "Expertise in a computer operating system: conceptualization and performance," *Human Computer Interaction*, Vol. 5, 1990, pp. 267-304.
[7] H. J. Einhorn, "Confidence in judgement: Persistence of the illusion of validity," *Psychol. rev.*, Vol. 85, 1978, pp. 395-416.
[8] I. Fischer and N. Harvey, "Combining forecasts: What information do judges need to outperform simple average?," *International journal of forecasting*, Vol. 15, 1999, pp. 227-246.
[9] B. Fischhoff, "Perceived informativeness of facts," *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 3, 1977, pp. 349-358.
[10] K. R. Hammond, *Human judgement and social policy*. New York: Oxford University Press, 1996.

[11] A. H. Hastorf and H. Cantril, “They saw a game: A case study,” *Journal of Abnormal and Social Psychology*, Vol. 49, 1954, pp. 129-134.

[12] S. J. Hoch and D. A. Schkade, “A psychological approach to decision support systems,” *Management Science*, Vol. 42, 1996, pp. 51-64.

[13] M. Jørgensen, “The quality of questionnaire based software maintenance studies,” *ACSM SIGSOFT - Software Engineering Notes*, Vol. 20, 1995, pp. 71-73.

[14] M. Jørgensen and D. I. K. Sjøberg, “Learning from experience in a software maintenance environment,” *Submitted to IEEE Transactions on Software Engineering*, 2000.

[15] D. Kahneman and A. Tversky, “On the psychology of prediction,” *Psychological Review*, Vol. 80, 1973, pp. 237-251.

[16] D. Kahnemann, P. Slovic, and A. Tversky, *Judgement under uncertainty: Heuristics and biases*: Cambridge University Press, 1982.

[17] M. R. Leary, “Hindsight distortion and the 1980 presidential election.,” *Personality and Social Psychology*, Vol. 8, 1982, pp. 257-263.

[18] G. B. Northcraft and M. A. Neale, “Experts, amateurs, and real estate: An ancoring-and-adjustement perspective on property pricing decisions.,” *Organizational Behavior and Human Decision Processes*, Vol. 39, 1987, pp. 84-97.

[19] S. Plous, *The psychology of judgment and decision making*: McGraw-Hill, 1993.

[20] P. M. Senge, *The fifth discipline: The art and practice of the learning organization*: Currency/Doubleday, 1995.

[21] W. Trochim, *The Research Methods Knowledge Base*, 2nd ed: Cornell Custom Publishing, Cornell University, Ithaca, New York, 1999.

[22] A. Tversky and D. Kahneman, “Judgment under uncertainty: Heuristics and biases,” *Science*, Vol. 185, 1974, pp. 1124-1130.

[23] A. Tversky and D. Kahneman, “Judgments of and by representativeness,” in *Judgment under uncertainty: Heuristics and biases*, D. Kahneman, P. Slovic, and A. Tversky, Eds. Cambridge, England: Cambridge University Press, 1982.

[24] R. P. Vallone, L. Ross, and M. R. Lepper, “The hostile media phenomenon: Biased perception and perceptions of media bias in coverage of the Beirut massacre.,” *Journal of Personality and Social Psychology*, Vol. 49, 1985, pp. 577-585.

[25] M. Jørgensen, D.I.K. Sjøberg, and R. Conradi, “Reuse of software development experience at Telenor Telecom Software,” European Software Process Improvement Conference (EuroSPI’98), Gothenburg, Sweden, 16 – 18 November 1998, pp. 10.19–10.31.

Managing Inexperienced Programmers by Better Managing Design-Coding

Kim Man Lui
Hong Kong Polytechnic University
Keith CC Chan
Hong Kong Polytechnic University

Introduction

The success of a software process relies on not only a well-defined paradigm but also people management. Technical programmer management remains an art form than an engineering principle. Many a software process does not take human characters, such as having the patience to solve problems and/or being loyalty to his company, into thorough consideration. As a result, some models work well for some software teams in some environments; however, they may not work as well for some others, especially those in developing countries where programmers are mostly inexperienced. There is a saying to bring right developers in a software team and/or to train inexperienced members as IT professionals. As IT is not as well-exploited in these backward areas, they do not have many opportunities to be involved in different development projects. To demand them to excel in their assigned tasks, some relatively mechanistic programming methodologies need to be developed

Our work was based on the experience we have gained over the last three years in managing a Corporation Information System (CIS) development project in an international premiere brewery located in a rural area in China. The whole development team consisted of members that had never programmed in a multi-user database environment before. As the demand for IT graduates in more developing areas in China was high, those that could be recruited in a poor rural area were usually inexperienced. In addition, many of them did not receive proper training in computing. Besides these, a rate of personnel turnover was typically high. As long as they received some training, many of them would seek a job with a better career prospect in more developed cities in China. This resulted in a vicious cycle where the project manager always had to work with programmers who were green and inexperienced. The situation of a high turnover rate was aggravated sometimes by resignation without any notice in advance. It happened several times that team members tendered resignation and chose to leave on the same day. Clearly, handing over of work was very difficult if not impossible and the team had to work under-staffed constantly.

To turn inexperienced programmers into contributors and make complex software more manageable, these technical management issues have to be dealt with [1]. They are highly related to the type of a software development paradigm chosen. One such choice is the object-oriented technologies. A number of self-contained classes can be built to hide complicated logic from the inexperienced developers. However, our work focuses on seeking or building knowledge patterns at a design stage. The patterns can then be embedded in written instructions or classes as tailored-made reusable components for the inexperienced programmers. Let us have two examples of the patterns.

The first one is concerned with inserting a row into a table. A person who knows syntax of inserting goods-received data into a warehouse table should have no problem with inserting invoice data into an invoice table, even though the semantics for these two business operations are quite distinct. However, coding the insertion of data into more than one table would be different. It requires the understanding of flow-of-control to take care of cases of a partial success.

Another of the pattern is to convert two different syntactic operations into one. Updating or deleting a record from a table can be combined into a pattern as we can transform deleting a row into updating a status column of deleted row of a database table. The deletion is thus logical rather than physical. As of (2) shown below, a flag column of the invoice table indicating the validity should be considered at a physical database design level. This is an example of how design affects coding. It has been suggested in [2] that data structure and control abstractions were generally best designed when they were done together, even though it is rarely done in fact.

(1) Delete an invoice AD135 from a database
(2) Mark an invoice AD135 as invalid in a database

This paper proposes a method known as MDC (<u>M</u>anaging <u>D</u>esign-<u>C</u>oding) to facilitate the co-design between data structure and control abstractions. We first deliberate how inexperienced programmers could make contributions, as MDC is for them. Then we will look at the knowledge patterns and logic templates, which are the groundwork of MDC. The general idea of MDC is to ensure that coding is planned ahead at the physical database design stage so that (1) complicated controls are eliminated from inexperienced developers and (2) the complexity and/or creativity required of coding is minimized by confining coding to a set of knowledge patterns. MDC can be considered as a programming environment in which programming still requires human effort but less of creativity. Developers are guided to concentrate on (1) how coding templates should be modified (the syntax) and (2) how visual objects should be assembled. With MDC, we have developed a number of multi-user database applications coded by inexperienced programmers and implemented in a brewery and four regional offices in China.

Inexperienced Programmers

One of the goals of knowledge management is to help employees within an organization to contribute more to their employer [3]. Even though the statement is clear, the way to achieving such a goal is vague indeed. One may argue that educating inexperienced people or allocating suitable jobs according to an individual’s ability may well fulfil the same purposes. However, when the knowledge and experience of staff members are not properly aligned with the tasks assigned, the learning curve is steep and long [4]. Nevertheless, when a staff becomes well-trained in some less developed regions in China, for example, his determination to look for better job prospects somewhere else in China will become stronger. Training, therefore, does not provide the best solution in this case. As for allocating developers according to their skill set, the idea is not feasible when all team members are inexperienced. Human resource allocation can therefore only be implemented with limited extent. Better knowledge management, other than adopting these principles, is required.

In recruiting and managing software developers, it is noteworthy that studies have shown that there are wide variations in productivity of programmers. Given the same specifications, the better ones can be 28 times more efficient [5]. Based on our experience again, there are also differences between “slow developers” and “unskilled developers”. They conjure up different images in the mind and it is subtle for their unrelated and independent differences to be pointed out. To gain as much from inexperienced developers as possible, we first need to clarify what we mean by inexperienced programmers in an engineering manner.

For a software project in MDC, capabilities and skills are classified into 3 groups: **BASIC**, **COMPETENT** and **REQUIRED**. The first one is about fundamental skills. Programmers equipped with these prerequisites alone are not enough to be ready for programming jobs in a CIS software project. The second is competent skills that are a complement to the group BASIC. Thus, from technical point of view, people having the two groups in total, so called the skills of experienced programmers, will be able to handle programming according to any workable design specifications. The definition so far is quite straightforward.

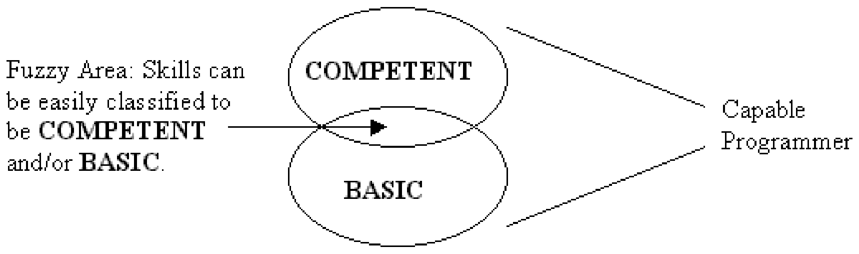

Fuzzy Area: Skills can be easily classified to be COMPETENT and/or BASIC.
COMPETENT
BASIC
Capable Programmer

Fig.1 BASIC and COMPETENT

The third group **REQUIRED** represents skills that are required by people having **BASIC** in order to be able to follow a pre-defined mechanism to reach the same results that, as if, qualified programmers achieved.

Among **BASIC**, **COMPETENT** and **REQUIRED**, the relationships can be best depicted by an example. Suppose you have a group of people, say mental defectives, who are able to perform number counting but do not understand mathematic addition. If we would like them to do the addition without a calculator, the most possible way seems to teach them the calculation. Again the learning curve may be long. Another approach to getting the same work done is by a mechanical style method that is to ask them to follow a predefined mechanism for counting marbles. As for 4+3, the rule may be as follows: (*step 1*) count four marbles and put them aside, (*step 2*) count three marbles and put them aside as to repeat the step 1, (*step 3*) mix it and count all the marbles.

Now, **BASIC** is “counting number”, **COMPETENT** is “doing addition”. **REQUIRED** is “knowing how to follow steps of counting stones”. Reusable components are marbles, a description of step 1 and a process of counting marbles. In this setup, we make the following claim: those people appear as perfectly understanding the addition calculation, since of course he makes the work done. However, in fact it is the approach effectuates the outcomes. One might argue that our rule is impractical for 811006+1, for example. In this case, we need to add more steps for improvements in effectiveness and efficiency. The metaphor is used to exemplify the philosophy of MDC for managing inexperienced programmers.

In this paper, an inexperienced programmer for database application development is defined if he/she has no experience in **COMPETENT** but covers **BASIC** and is able to learn **REQUIRED** by on-the-job training. Based on our experience, Table 1 shows these three groups of different skills for relational database development in MDC.

COMPETENT	1 Handling transaction, 2 Handling control of failure, 3 Handling object class, 4 Handling concurrency, 5 Deleting rows from database, 6 Coding subsystems from scratch, 7 Writing precise comments inside source code, 8 Considering logic with boundary values

BASIC	1 General mathematical logic, 2 IF-then-else and while loop, 3 Procedure and function, 4 Local and global variables (but not instance variable, 5 Four variable types: string, integer, decimal and datetime
REQUIRED	1 Mimicking patterns from logic template, 2 Finding a right event from visual objects in windows programming, 3 Remarking a revised date and name of developers inside the code for version control, 4 Coding SQL statement by select and insert and update, 5 Coding SQL statement by where and sub-query

Table 1: Three Groups of Support Knowledge

By managing design-coding, we aim at defining mechanistic rules or templates for a software process in which programmers with **BASIC** and **REQUIRED** could do programming. With **BASIC** as the basis, however, the team must be able to attain the skills of **REQUIRED** through on-the-job training. This type of training involves both sequential and concurrent teaching. **REQURIED** knowledge exists a priori in MDC but it must be learned and the learning process involves people being supervised in their work [6]. MDC provides guidance during that process. To follow it, the team needs to acquire **REQUIRED** by practice. The guidance is to define how to do programming tasks based on two skill sets **BASIC** and **REQUIRED**. For the above example of counting marbles, it is step 1, 2 and 3. In abstract, the guidance knowledge from MDC is written in **BASIC** and **REQURIED**.

There are two types of knowledge to manage [3]. The first type is support that serves as a basis for understanding of what; the second is guidance that provides explanations of how to accomplish a task. The use of the guidance is to deliberate how the support knowledge is applied. Guidance depends on support knowledge. Without providing relevant the support knowledge, the guidance becomes meaningless because it is used to answer how to make best use of a specific kind of support knowledge. From a knowledge perspective, **BASIC**, **REQUIRED** and **COMPETENT** pertain to the support knowledge. MDC provides the guidance to two sets of the support knowledge, **BASIC** and **REQUIRED**. We argue that a good paradigm in software development is the function of guidance used to go through the process.

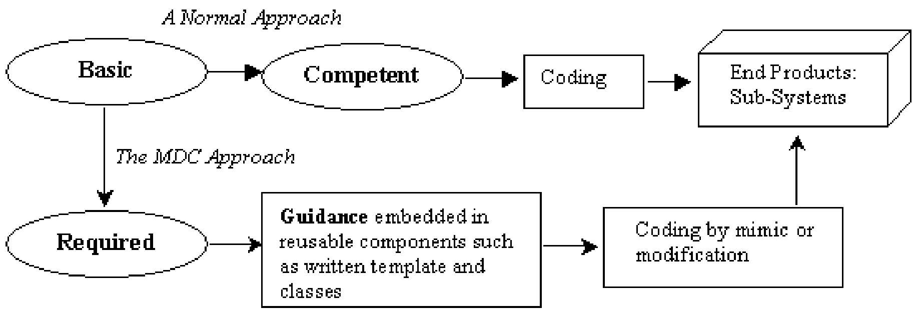

A Normal Approach
Basic
Competent
Coding
End Products: Sub-Systems
The MDC Approach
Required
Guidance embedded in reusable components such as written template and classes
Coding by mimic or modification

Fig. 2: Application of Three Groups of Support Knowledge

This section is critical. First, the minimum requirements of new recruited staff are **BASIC**, which is easier to meet. Second, knowledge patterns are confined to **BASIC** and **REQUIRED**. It then lessons potential impacts of handover, as there is no coding technique created from a developer that others cannot follow. Ideally, people can simply join and work. Third, coding productivity increases because more people are qualified to do programming and to be able to follow any other people's work in short time in the MDC environment. In short, this directly correlates with inexperienced programmers, a personnel turnover and productivity. The next section will describe the syntactic pattern, which is the groundwork of MDC. Based on this, MDC can formulate the guidance being appeared in various forms such as a data model, a blueprint of ways to programming and logic templates, in which customizations and applications of them require the use of **BASIC** and **REQUIRED**.

Coding Pattern and Logic Template

Programming database transactions is the key part of a software process for CIS development. From a complexity perspective, ANSI SQL II statements in a database transaction can be expressed as a tuple that consists of two components: type of operations (i.e. retrieve, insert and update) and number of tables involved in a single statement. As mentioned, a technique for inserting records to a single table can be re-applied to any other single table. It employs the "insert" operation and involves one table, e.g. insert into table_x (col_a, col_b) values (123, 'abc'), meaning that inserting a record 123 and 'abc' to two columns col_a and col_b of table_x respectively, denoted as (i,1).

(i,1) is regardless of semantics and thus is independent of application requirements. Of inserting a trading term into an invoice table or a product quantity into an inventory table, the complexity remains the same and (i, 1) is a pattern of coding.

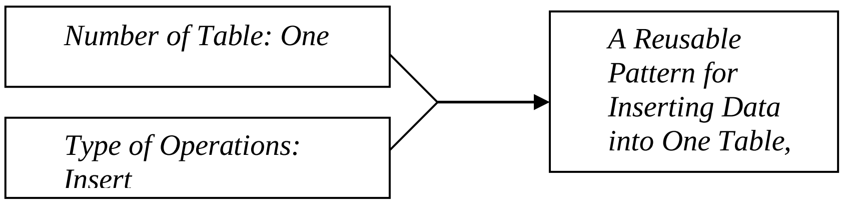

Number of Table: One
Type of Operations: Insert
A Reusable Pattern for Inserting Data into One Table,

Fig. 3: A Reusable Pattern

On considering a compound statement such as "insert into table_x (col_a, col_b) select col_p, col_q from table_y where col_p = 123", we denote this kind of a coding pattern as ((i, 1), (r, 1)), where (i, 1) is a leading operation and the tail is (r, 1) one level sub-query. Obviously, coding ((i, 1), (r, 1)) is more difficult than writing (i, 1). Thus the level of a query controls the complexity as well.

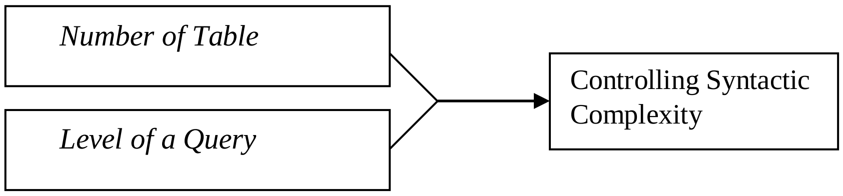


Fig. 4: The number of table and level of a query in an operation controls the syntactic complexity.

Should a database be designed in a single table, many of transactions could be manipulated in (i, 1) and (u, 1). The programming would be just simpler in syntax than ever, but also clumsier and more inflexible. Thus, our attempt is to balance the syntactic complexity at the programming level by managing the database design. Two magic numbers, the maximum number of tables and the maximum level of a query involved in any single statement, determine the whole complication of programming works. This should then be carefully considered at the physical database stage, i.e. coding planned ahead in designing. In addition, the set of the patterns, i.e. (i, 1) and (u, 1), is a focus of building reusable components for inexperienced programmers.

Few of database applications are built just by pieces of unlinked statements. For example, we design invoice tables as master and detail, rather than one single invoice table. Thus we need to add a new invoice into both two tables by two (i, 1) operations, which may cause an integrity problem, i.e. both the operations must either fail or succeed. If the data integrity is resolved systematically, database designers will be able to apply table redundancy for performance [7, 8] and, more importantly, temporary tables for the optimization of programming complexity. We now introduce a logic template that sets up a frame of control flow, which can be embedded in normal function calls and/or class functions. What programmers do is to mimic the structure of a pattern. A sample of the logic template for data integrity may look as:

```
Begin transaction
    Update table_1 set col_a = date_add(getdate(),+1-1,d)
    /* col_a = today = getdate() + 1 day – 1 day */
    If rowcount<>0 and @@error <>0 rollback transaction
    Insert table_2 select col_a, col_b from table_1
    If rowcount<>0 and @@error <> 0 rollback transaction
Commit transaction
```

By no mean, inexperienced programmers digest how the above template protects all works of a transaction for being either successful or failed. They should only view the statements in bold.

```
Update table_1 set col_a = date_add(getdate(),+1-1,d)
Insert table_2 select col_a, col_b from table_1
```

For new joined programmers, they concentrate on their code semantically correct. The implementation can be done by mimicking SQL statements from logic templates. Furthermore, the template could be designed as self-explanatory. For instance, although the update statement is equivalent to "Update table_1 set col_a = getdate()". When it is yesterday, programmers require to know a syntax for "getdate() – 1" which is semantically right but syntactically wrong. The design of the logic template should point at the capability of a particular software team; it should not go theory but practice. From our experience, it was interesting to note that feedback from inexperienced programmers would make good improvements for the self-explained in logic templates because it might be hard to imagine why they cannot follow the templates.

The logic template is used to control a transaction for commit and rollback, which was just elaborated. It also ensures the achievement for the flow-of-control. The templates are guidance knowledge to programmers. Through the logic templates, the inexperienced programmers revise some statements that require the skills of **BASIC** and **REQURIED**. As mentioned, modifying the templates requires two sets of support knowledge **BASIC** and **REQUIRED**. Our mechanism provides a virtual capability to the developers so that they could then do tasks that should require the knowledge of **COMPETENT**. This is our augment that complicated controls are transparency to the developers. The programmers now focus on semantic meanings of a set of SQL statements, but not the control of linking among them. Finally, in practice, it is faster to document a logic template with modification parts in bold where should be revised for customization; anything not in bold should be copied exactly. The developers then complete tasks by following it.

It is noted that the relationships among database design, coding pattern and logic template implemented in either objects or procedure calls are intertwined and should not be considered separately. By analogy, lung and heart are two organs in function, but both equal important to the cycle of a human oxidative process.

Managing Design and Coding

In the previous section, we presented two magic numbers: the maximum number of tables involved in a SQL statement and the maximum query level of a SQL statement. Considering these at design, we limit the degree of syntactic complexity in programming. Also the logic template was proposed for coping with a control of transaction. This section will discuss a software development process of managing design-coding in which we gain the achievements of implementation of logic templates discussed in previous section.

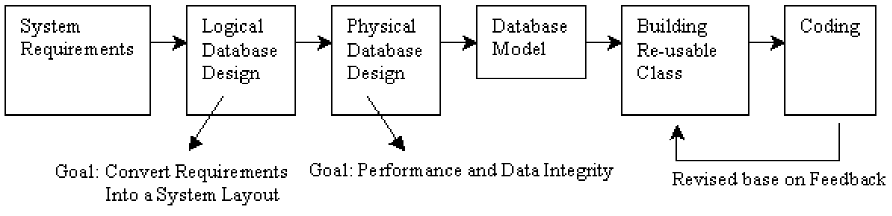


Fig. 5: Contemporary Software Development

Traditionally, one of design goals for the physical database tends towards performance improvement through de-normalisation, table partitioning and indexing [9]. Therefore, we have to ensure data integrity for accuracy of information, especially when data is de-normalized or re-configured in a way of fulfilling the required performance for OLAP and/or data warehousing [10]. Thus, database architects do not consider the design and programming altogether, even though there are implications between them. The very fact is that a professional software team is able to program systems irrespective of any database architecture. Consequently, the practitioners keep being interested in making a data model for machine efficiency because a poor design would post limitations of tuning and could eventually incur a high cost for maintaining performance. After the design is complete, object classes related to a database model can be defined for reuse. Different parts of the model might have huge differences in their data manipulation and coding implementation so that an object class to one part could not be re-applied to the other. This will result in having more classes.

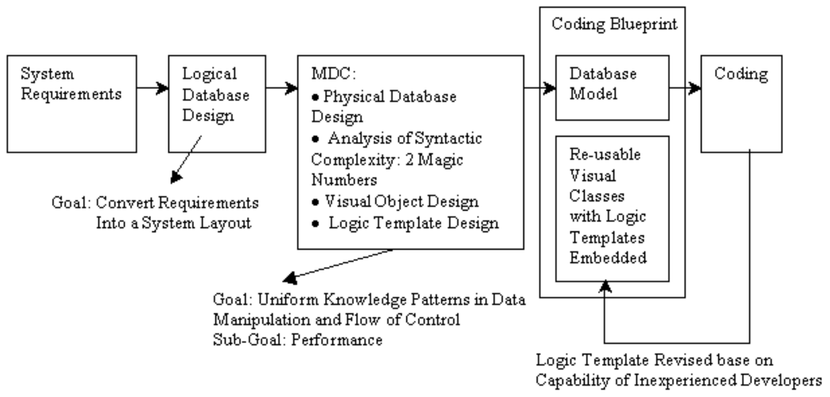

System Requirements
Logical Database Design
Goal: Convert Requirements Into a System Layout
MDC:
• Physical Database Design
• Analysis of Syntactic Complexity: 2 Magic Numbers
• Visual Object Design
• Logic Template Design
Goal: Uniform Knowledge Patterns in Data Manipulation and Flow of Control
Sub-Goal: Performance
Coding Blueprint
Database Model
Re-usable Visual Classes with Logic Templates Embedded
Coding
Logic Template Revised base on Capability of Inexperienced Developers

Fig. 6: Managing Design and Coding

Fig 6 illustrates the process of management of design-coding. In comparison to Fig 5, the objective of physical database design is to seek a model that data operations in terms of coding patterns are consistent in syntactic logic as many as possible. One of the functions of a logic template is to link up these patterns such that flow-of-control among them is embedded in the template and thus eliminated to developers. Performance becomes a sub-goal. In addition, the logic templates are often to be revised for easier to follow. Unlike Fig 5, we could reveal a precise statement to indicate when revisions of object classes are needed.

The physical database design is merged with the syntactic analysis, visual objects and the logic template. The syntactic analysis is critical. For instance of stock in and out, we can create a table with a quantity column in which values can be positive and negative representing in and out, respectively. Another way is to have two tables recording in and out, respectively. Either of the ways is not judged as a superior. In MDC, it is justified in accordance with that other part is taken the same approach so that knowledge patterns are reused as many times as possible. If we use two cash tables for inflow and outflow, respectively, then designing the two tables for stock in and out might be better in a sense. It is notified that employing two tables for in and out, instead of one, makes coding prompt to error for inexperienced programmers because many operations need to be applied to both the tables. Through the logic template, the integrity issues can be resolved. We are more concerned about the maximum number of the table involved and of the query level, instead of the number of SQL statements in a transaction and its flow of faulty control.

The design goal becomes to make as many as data manipulation into a set of common knowledge patterns so that programming can be confined to them. In order to achieve this, we might make one table into several tables or some temporary tables. Therefore, the number of SQL statements in a transaction increases and thus we require the logic template for the transaction control. The implementation of logic template can be inside visual objects. For example, we might have (i,1) and (i,2) operation-oriented visual object classes. When a developer copes with (i,2), he will use the corresponding visual objects. From this point, our visual classes have a hierarchy of programming properties, instead of the hierarchy of application objects such as People -> Employee/er -> Manager -> Clerk. Therefore, visual classes in MDC are concrete programmable components. They are not of application dependence, and thus their reusability is maximized.

The design goal of MDC is clear and the considerations of the design are also well defined. The feedback for revision again is precise. What patterns should be sought and embedded in a logic template has been discussed. We also linked MDC with the capabilities of the inexperienced programmers.

Conclusions

The work was practiced in an in-house IT software team in China. During 1997 and 98, a major system called New Sales and Distribution System (NSDS) was first developed and implemented in a brewery and regional sales offices. NSDS was also the first practise of MDC. During the development, a rapid personnel turnover happened as expected. However, new recruited staff would be able to follow without any major setbacks. Around the middle stage of the development, the IT manager was involved a little in programming. By that time, a set of logic templates and visual objects were becoming sophisticated. The logic templates were modified in order to make them become the effective guidance to local staff. From our experience, new joined staff would be able to contribute to system development after two-day on-the-job training. Afterwards, Capital Expenditure, Marketing Database, Logistics Management and Equipment Registration were then developed using the same approach. It was noted that the manager became passive after NSDS because local staff can pass their understanding of the mechanism of MDC to new staff.

MDC resembles a production line. Once it is designed and running smoothly, its architects are no longer in need. Monitoring the operations of the line is nothing technical, so as MDC. In MDC, an individual developer plays a labour-wise role and his skill sets required are not demanding. What we really need is to ensure the people following a mechanism, not to breaking any rules or to create his own style of coding. Applying support knowledge and guidance for software development, we are brought to a mechanical programming environment in which tasks are well defined and people are confined to the guidance for completing their jobs. This is similar to the production assembly.

In terms of knowledge management, contrary to our approach to software development, another one is knowledge factory [6]. It is common nowadays although the term knowledge factory is rarely used in software industry. During development, programmers actually precede two processes, learn and work, concurrently. Thus the software team is exploring innovative ways or constructing new knowledge, which is called knowledge factory in a sense. As for MDC, the knowledge of programming development, i.e. support and guidance, exists a priori. In contemporary software engineering, not of the programming development knowledge exists a prior but there are always new experiences and knowledge that is being constructed by the software team during development.

For comparison, we collected feedback from some ex-team members. Two of them worked for another development project. They were trying to apply the MDC reusable components such as logic templates or sub-modules for their new project. However, the project leader once blamed them for the progress. He believed that they had no basic sense in computing although they could show themselves knowledgeable to user requirements. They were even questioned how they did the jobs before. This resulted into all modules related to data manipulation rewritten by other developers from Hong Kong. It was then unable to meet the project deadline. Those ex-team members referred that the database design was different from the ones they worked on. Thus, they made some changes that should not be changed in the logic templates in order to fit their database design. Once modified, there existed many hidden technical issues belonging to **COMPETENT** that happened later and could turn out more revisions of the templates or classes. The problem explained that the integration of the database design, magic numbers, visual classes and logic templates should be seamless, and cannot be used individually. It also showed that the semantic reusable and syntactic reusable could be independent of each other. The logic templates and visual objects in MDC are syntactic reusable only if a database is designed with coding planned ahead. Although those programmers got familiar with **BASIC**, **REQUIRE** and some of **COMPETENT** in our software environment, which are support knowledge. Without guidance knowledge, a programming process will leave programmers' talent to judge how the support knowledge is employed.

This paper might become arguable, as it is just a method of software development for inexperienced programmers. Therefore, its application to advance the technology has not been easily recognized. However, towards a global software team, our work will provide a good reference for mechanism-oriented programming in remote. Also, around-the-clock development requires an extreme mechanism-oriented paradigm; otherwise, a west-sphere team is not able to catch up with what the progress has been made by an east-sphere one in a half hour.

Finally, little part of the work was published in a Chinese software magazine in PRC [11]. The authors would like to thank Richard Messnarz and Tor Stålhane for their helpful comments.

References

[1] Ropponen J. and Lyytinen K. Components of Software Development Risk: How to Address Them? A Project Manager Survey, *IEEE Transactions on Software Engineering*, pp. 98-111, February 2000

[2] Gabriel R.P., Patterns of Software, Oxford University Press, 1996

[3] Wilson L.T., Snyder C.A., Knowledge Management and IT: How Are They Related, *IEEE IT Pro,* pp. 73-74, March 1999

[4] Amrine H.T., Ritchey J.A., Moodie C.L., Kmec J.F., Manufacturing Organization and Management, Prentice Hall, 1993

[5] Humphrey W.S., Managing Technical People, Addison-Wesley, 1997

[6] Silveira M.A. and Scavarda-do-Carmo L.C., Sequential and Concurrent Teaching: Structuring Hands-On Methodology, *IEEE Transactions on Education,* pp. 103-108, May 1999

[7] Roy S. and Sugiyama M., Sybase Performance Tuning, Prentice Hall, 1996

[8] Gurry M. and Corrigan P., Oracle Performance Tuning, O'Reilly, 1997

[9] Martyn T., Implementation Design for Databases: The 'Forgotten' Step, *IEEE IT Pro*, pp. 42-49, March 1999.

[10] Chaudhuri S. and Dayal U., An Overview of Data Warehousing and OLAP Technology, *ACM SIGMOD Record*, pp. 65-74, March 1997

[11] Lui K.M. and Zhang J., Software Development in China (written in Chinese), *Electronics and Information*, Beijing, China, pp. 26-27, March 1999

The Hong Kong Polytechnic University

The Hong Kong Polytechnic University is a world class university offering both academic and professional education of the highest quality. For over 27 years, the university has been internationally recognised for academic excellence and extensive research activities. The research direction of the university is application-oriented and thus is an applied nature relevant to industrial, commercial and the Far East community needs.

The Department of Computing is currently working on 91 funded research projects from Hong Kong government and the commercial industries both in Hong Kong and overseas. In addition to these, the Department also involves a number of theoretical studies in computer science. On average, the Department published 50 international conference and journal papers annually.

Author : K M Lui

Kim Man Lui received a B. Eng in 1990 from Tamkang University, Taiwan. He was awarded a bursary to study for his MSc in computer science at the University of the Witwatersrand, South Africa. There he received the MSc in 1994. He was invited to be the chairperson for the hardware session of the Asian Conference on Computer Vision 1993 held in Japan. Afterwards, Mr Lui has worked in a number of IT positions in the commercial sector from system engineer, analyst programmer, system analyst, project leader and IT manager. Mr Lui was a certified Sybase Database Administrator in 1996 and also a certified Oracle Database Administrator in 1999. While studying towards his PhD degree at the Hong Kong Polytechnic University, Mr Lui is currently working as the IT manager for Carlsberg responsible for Pan China IT development and support. Mr Lui published several international journal and conference papers on pattern recognition, intelligent control, qualitative reasoning and knowledge discovery. His current interests include software engineering, technology-based project management, knowledge management and remote software team.

Author : Keith C.C. Chan

Dr. Keith Chan has a B.Math. (Hons.) degree in Computer Science and Statistics and a M.A.Sc. and Ph.D. degree in Systems Design Engineering from the University of Waterloo, Ontario, Canada. He has a number of years of experience in software development and management. Before joining The Hong Kong Polytechnic University, he was with the IBM Canada Laboratory where he was involved in every phase of software development from design to coding, to testing and maintenance. He was a member of the software quality assurance team during the time the laboratory first applied for ISO 9001 certification.

Dr. Chan has been working on different areas in software engineering for many years. He has successfully led the effort to develop a self-assessment tool for ISO 9001 certification and a workflow automation tool to facilitate the adoption of ISO 9001 compliant practices. Dr. Chan has conducted courses and seminars in software quality management, and software project management. He has published in the areas of software processes, software quality management, multi-site software development and cooperative information systems. He has been a consultant to government, semigovernment and commercial organizations in Hong Kong, Canada, Singapore and Malaysia. Dr. Chan is a reviewer for IEEE standards in software engineering and is a founding member of the IEEE Standards Association.

Dr. Chan joined the Department of Electrical and Computer Engineering at Ryerson Polytechnic University, Ontario, Canada as an Associate Professor in 1993, before returning to HK to join the Hong Kong Polytechnic University in 1994. He is currently the Director of the Software Technology Facilities Center and an Associate Professor of the Department of Computing at the university. Since August, 1999, he has been an Adjunct Professor of the Institute of Software, the Chinese Academy of Sciences, Beijing, China. From 1992 to 1996, Dr. Chan was an adjunct faculty member in the Department of Systems Design Engineering, University of Waterloo, Ontario, Canada, the Department of Electrical Engineering, The University of Western Ontario, Ontario, Canada, the Department of Electrical and Computer Engineering, Ryerson Polytechnic University, Ontario, Canada.

Dr. Chan was the Program Co-Chair of the *First and Second Hong Kong Conference on Quality Software Development* in 1996 and 1997. He has been involved in several software process improvement and ISO 9000 projects in Hong Kong and China. He is a trained ISO 9001 and TickIT Assessor.

A field-study of Cultural Influences on Software Process Improvement in a Global Organisation

Kerstin V. Siakas [1,2] and Bo Balstrup[3]

[1] University of North London
School of Informatics and Multimedia Technology
2-16 Eden Grove, London, N7 8DB
Tel.: + 44 171 753 3142, Fax: + 44 171 753 7009
E-mail: k.siakas@unl.ac.uk

[2]Technological Educational Institution of Thessaloniki,
Department of Informatics, P.O. Box 14561
GR-54101 Thessaloniki, Greece
Tel: +30 31 791 296, Fax: +30 31 799 152
E-mail: siaka@it.teithe.gr

[3]Danfoss Drives A/S
DK-6300 Graasten, Denmark
Tel: + 45 7488 2222, Fax: + 45 7488 5445
E-mail: balstrup@danfoss.com

Abstract

The general aims of this research are to find out how culture influences the successful implementation of Software Quality Management systems. The hypothesis is that both the national culture and the organisational culture affect the take up of Software Quality Management Systems and the awareness of the workforce for quality issues. It is especially important for global organisations with virtual cross-national teams to understand the importance of diverse individual work values. The objectives of global organisations are to create a universal culture in the whole organisation with converge values and simultaneously to adapt to the national cultures in which they operate allowing divergence behaviour in order to achieve high business performance.

A field-study was conducted in a global organisation during February and March 2000 at three divisions in their software developing departments. During the field-study a maturity assessment took place in one of the divisions. The aim of the field-study was to gain deeper understanding of the cultural aspects relating to software quality. Observations and totally 56 interviews with employees in software development on different levels were conducted. Before the interviews the participants had answered a questionnaire developed by the author. The assessors were also interviewed, as well as two employees working with a SPI project on overall corporate level. The findings intend to explain the national differences found in work values between three countries by using Hofstede's four dimensions and their effect on the awareness and take-up of the software development process in place. The organisational culture will also be taken into consideration and be analysed with the same purpose.

Introduction

About the organisation

The organisation is a global industrial organisation with about 20,000 employees around the world. It belongs to the leaders in research, development and production of mechanical and electronic components for several industrial branches. The products and services are used for vital functions in homes, at workplaces and in institutions - in fact in every human environment. The organisation is a global enterprise, with a reputation for using advanced technology in products and processes and for consciousness of the environment. All factories are or will be certified according to ISO 14001.

Divisions / Business Units

The divisions are business units inside the overall organisation. They become more and more separated. They buy and sell services to external customers as well as to other divisions inside the organisation. In this paper the findings from one of the divisions participating in the field-study is reported and will be referred to as the organisation, while the entire organisation will be referred to as the corporation.

ISO9000:2000

The overall corporation is ISO9001 certified on production level. There is no ISO 9001 certification especially for Software, but Software is embedded in the final product, so indirectly it also applies for Software.

A revision of ISO9001 is made to apply to the new ISO9001:2000. The new structure is process-oriented and based on the Plan-Do-Check-Act principle. It replaces the old ISO9001 to ISO9003 and is aligned with ISO14000. The major clauses are management responsibilities, resource management, product realisation, measurements, analysis and improvements.

Corporate SPI

A SPI project was established a few years ago on corporate level for Software Process Improvements. The idea came up on the cross-divisional meetings, which usually were held once a month. The initial intention with the corporate SPI project was to hire two persons dealing with software process issues together with representatives allocated to the project from the different software developing divisions. The corporation financed the project by 50 %. Practically this cross-divisional co-operation did not succeed, because of difficulties with allocating people to the project and different maturity levels in the divisions. Instead the corporate SPI project started to work as consultants for the divisions interested in Software Process Improvements. A system was developed, called Software Process

Assistance (SPA), which is available for all the divisions. It is a web based tool including process descriptions, roles and recommended templates to be tailored for individual software development projects.

National culture

Hofstede [Hofstede 1994], who's framework has been widely accepted and cited in the cross-cultural management literature, undertook in the 1960's extensive research about work-related cultural values in a major multinational firm spanning 50 different countries. The principal difference among the respondents was culture; all of them were otherwise similar because they were carefully matched for other characteristics such as age, sex, and job category.

The findings from Hofstede's research are four independent dimensions of values:

- *power distance*, which describes the extent to which hierarchies and unequal distribution of power is accepted
- *uncertainty avoidance*, which indicates the extent to which a society feels threatened by ambiguous situations and tries to avoid them by providing rules, believing in absolute truths, and refusing to tolerate deviance
- *masculinity versus femininity*, which describes the relationship between the masculine assertiveness, competitiveness and materialism opposed to the feminine concern for quality of relationships, nurturing and social well being
- *individualism versus collectivism*, which describes the relationship between the individual independence and the collective interdependence of a group

Table 1 shows Hofstede's [1] work-related values for the three countries in which the organisation in this field-study develop software.

	Denmark	**US**	**Germany**
Power Distance	18	40	35
Uncertainty Avoidance	23	46	65
Masculinity / Femininity	16	62	66
Individualist / Collectivist	74	91	67

Table 1. Work-related cultural values according to Hofstede [1].

The opinion of the software developers about differences in national culture

Common views held by the interviewees are reported about the three national cultures in which the organisation develops software. The authors have grouped the views according to Hofstede's four dimensions [1]. The views are seen from an organisational point of view and thus biased. The views reported cannot be considered epistemologically valid, but they are an indicator for cultural differences influenced by the national culture.

The Danish Culture

Power Distance (18, low power distance)

- they are egalitarian (the sense that every person is equal)
- they are conservative and not very innovative

Uncertainty Avoidance (23, low uncertainty avoidance)
- they work effectively and are very engaged in their work, but on the same time easygoing
- they take responsibility more seriously
- they are planners and very schedule conscious
- they are very structured and like to follow procedures closely
- they are not very willing to take risks

Masculinity / Femininity (16, one of the most feminine countries)
- social well being and family come first
- they have a softer relationship with each others

Individualist / Collectivist (74, individualist)
- the process is very slow, because they have to discuss everything with their colleagues
- decision making is a slow and delegated process

The US Culture

Power Distance (40, medium power distance)
- there is a big hierarchy
- decisions are usually taken by the boss
- the discussions are usually dominated by the boss
- communication usually has to go through the boss

Uncertainty Avoidance (46, medium uncertainty avoidance)
- they need instructions and direct orders
- information is not shared, because it is power
- they are willing to take risks

Masculinity / Femininity (62, masculine)
- time is money, everything has to be big and fast (short term focus)
- the communication style is indirect - many things are said between the lines

Individualist / Collectivist (91, the most individualist country)
- they are doers, they have a need to prove they are productive
- they are very competitive
- they work alone and do not ask for help easily - they like to close themselves into small boxes
- they are very self reliant and they do not discuss solutions to any higher degree
- team work is dividing the work into smaller tasks, not working together
- the attitude is to work until you finish your job
- they blame others more easily if they fail
- nobody trusts anybody, which means that nobody has access to anything
- they do not stick their neck out, but they ask a lot of questions

The German Culture

Power distance (35, medium power distance)
- they are used to have the boss telling them what to do
- the decisions are usually taken by the boss

Uncertainty Avoidance (65, high uncertainty avoidance)
- they do no question things, they do as they are told even if they do not like what they have to do.

Usually afterwards it is found out that they were not satisfied.
- they do not ask much
- they are more risky than the Danes

Masculinity / Femininity (66, masculine)
- they are very task oriented

Individualist / Collectivist (67, individualist)
- they are doers
- they are competitive

National differences found during the field-study

The Danes

The Danes seem to be very easygoing and taking their roles seriously. Their punctuality for the interviews meetings was impressive. They are proud of working for the organisation and they are committed to what they are doing. There is no competition and everybody seems to willingly both ask for help if needed and also proud of being able to give help if asked for. They work well in teams and they are empowered to take delegated decisions.

The Americans

The Americans were very keen on stressing that they are ambitious, hard working and like to be productive. The structure is more hierarchical and competitive. They feel they do not have enough information from the mother organisation about what is going on and about the future. They would also like to take a more active role in the change process and in decision making, especially about tools and technology. They think that the Danes are somehow conservative, not willing to take risks and not very innovative. They are very individualistic, but they also want in development to feel like a team. Trust, seems to be important for collaboration. They would like more focus on target market issues and focus on being in the leading edge. They want things to happen faster, technology to change faster and to push the limits. They think time to market is too slow. They would like to have feedback earlier in the developing process and they would like to start coding earlier. They also want acknowledgement for their efforts and respect from colleagues in Denmark as equal professionals. They think management in Denmark is not aware of their professional level and of how ambitious and good professionals they are.

The Germans

The organisational structure in Germany in general consists of a taller hierarchy than in Denmark. This means that the span of control is narrower with more levels between those in the bottom and the top. The Germans are competitive, task-oriented and in general committed to the organisation philosophy. They feel they need more instructions or direct rules about how to work. They also feel that they do not have enough information from the mother organisation in general and that there is a need for training especially in quality issues. They have a good team spirit and they seem to accept things they don’t like more easily than the Americans do. They want to be acknowledged for their achievements and they want to be respected as equals by the Danes.

Organisation Culture

The organisation has become a global organisation with software development in Denmark, Germany and US. Most of the software developers in the organisation have an electronic background and have been working for many years in the organisation. The software products are embedded control software.

The Core Team

A couple of years ago a re-organisation was made. One of the results of this re-organisation was the creation of Core Teams. The Core Team is headed by a project manager and includes members from all relevant departments in the organisation. From the project group usually the project team leaders take part. The Core Team handles questions of importance for the product development and takes decisions on a higher level, especially decisions of economical importance.

The Software Quality Assurance Department

In order to set focus on software quality and SPI, when software became a core process, the Software Quality Assurance Department was established in the organisation. Today the Software Quality Assurance Department consists of 4 persons. The department is responsible for the process improvements and for controlling that the processes are followed. The Software Quality Assurance Department has taken actively part in the corporate SPI project, which has developed the Software Process Assistance (SPA). It has been possible to concentrate on the improvements important to the organisation, because the organisation had been the most active in the corporate SPI project. The templates developed in the SPA have been used in all new projects. The Software Quality Assurance Department is not a part of the projects, but supports and monitors the process. It also collects metrics from reviews and inspections. In US and Germany the local software manager is responsible for that the processes are followed.

The Quality Department

There is a Quality Department on a higher level, which is an integral part of the projects as a member of the Core Team. The Quality Department is responsible for divisional standards, ISO 9001:2000 and ISO 14000 compliance. The Quality Department assesses the quality produced by the projects and controls the quality of the products before release. The Quality Department and the Software Quality Assurance Department co-operate.

The individuals in the projects

The project itself and the individuals within it are the main responsible for the quality of the project (that the time schedule is followed) and for the quality of the product. Every week a project meeting is held where current status is given, problems are discussed and action plans are made. One person from the Software Quality Assurance Department takes part in the weekly project meetings in order to follow up on the process.

Maturity assessment

The organisation was maturity assessed four years ago. After that process improvements have been made mainly through active participation in the Corporate SPI project. The re-organisation introduced full quality responsibility within the projects and a rotation program where software developers move to the support department for two years. These changes in the organisation has promoted software process improvements.

A second maturity assessment took place in February 2000. There is still a way to go to reach level 5, the final goal, but the software process improvements are sustainable, and the organisation will concentrate on the weak areas to increase the maturity level. A short-term goal is that every project has to improve in at least one weak area pointed out at the assessment.

The opinion of the authors is that the workforce is loyal to the organisation and committed to the improvements. The process is supported by management, who is willing to assure the necessary resources and funding for the improvements.

The software developers viewpoint about issues relating to software quality issues

Software Quality

In the embedded world software defects are more crucial than in traditional software, because if software fails the whole product fails. Defects cannot be accepted in the production The process description, the documentation, and recording of defects have improved software quality. There is an emphasis on functional requirements, because the projects are measured on functionality. There are views that following the process description is not enough for software quality. There are other issues like the architecture and the internal programming structure that also add to software quality, but which usually are forgotten, because they are not measured. A software quality attribute like changability and traceability are examples of attributes originating from the internal programming structure. It is a question of business strategy. Is it important to have a flexible product adaptable to the market or a standard product that will not be changed for 5-6 years?

Another view is that the quality system is very heavy and not suitable for small projects. The organisation is still experimenting and there are opinions that not enough resources are put into software quality issues. The way of working is not the same in all projects. In some projects they follow the process by the book, while in other projects they use the process as a guide.

ISO9001 Certification

The official quality system for the whole corporation is ISO9001. The ISO9001 certification is for the whole of the production process. The quality department consists of people with background in hardware and production and with a limited knowledge of software. The ISO9001 quality system is seen as very distant from the software engineers’ point of view, because ISO9001 does not address software adequately and software is not emphasised by the quality department either. The ISO9000:2000, which addresses software issues to a higher degree, is replacing the ISO9001 and ISO9000-3, which are guidelines for software.

The software produced in the organisation is invisible to a high degree, because it is embedded in the final product. The ISO9001 review process concentrates on the total process and software is considered as a black box. The integration between SPI and ISO9001 is not clear and most of the software developers are not even aware of the quality manual required by ISO9001.

The corporate SPI project

The corporate SPI project, the basis for the process improvements, is created mainly outside the organisation on corporate level. Some of the software developers do not feel that it is really their system. The organisation has been the most active part in the corporate SPI projects and the improvements are created by the software engineers themselves. The inspection process for example was created inside the organisation. The corporate SPI project is not an official quality system in the organisation. It is still under development. Today it is used officially only in one project. The recommended templates in SPA include instructions and a checklist. There is a lot of information inside the SPA. The projects are supposed to use what they need and also to do improvements. No training in the use of the templates has been offered in advance to the software engineers, because it was considered that the software engineers first need some experience in using the templates and afterwards there can be discussions about training and improvements. This has been confusing and also created a negative attitude. The software engineers in general want training before using the SPA and they also want to see more strictly what to use, because in the projects they do not have the time to experiment with the templates.

There is a mixture of positive and negative attitudes towards the SPA system. Software developers who have the negative attitude feeling to a high degree that there is too much documentation, the wrong kind of documentation, too much writing of the same things, no consistency in documentation in the different projects and also difficulties with the levels of documentation. In some projects the documentation is too detailed and in some too broad. Mainly the overview is missing, while the documentation of a module seems to exist. Most of the developers feel that they don't know whom they are writing the documentation for. The value is not comparative with the time required to produce the documentation. They believe that nobody ever will read their documentation. It is very de-motivating to write all the documentation, but they do it just because they have to. The people with a positive view think that in a big organisation it is important to have a structure and to have formal ways of doing things. They believe that the templates are good and they use them as they are. Software developers who have been in the rotation system and have been working in the support department see clearly the advantages of documentation. They have a wish for higher level documentation and especially answers to the question 'why'. The author's impression is that there is some resistance against the documentation mainly dependent on the lack of seeing the advantages. Reuse today is carried out mostly by copying code or documentation. Real reuse is not used to any higher degree.

The Software Quality Assurance department

The Software Quality Assurance department is seen by many software developers as some guys sitting in a tower, far away from the real problems in the projects. To bridge this attitude the software developers have to participate in the change process. The software quality assurance department does not have any formal authority. It can only recommend or give warnings.

The software developers viewpoint about issues relating to organisational culture

Management commitment

The top management in the division is committed to the process improvements and also has awareness that they need to have a strong belief that the organisation will gain in the long term. In the projects there is a view that higher management is committed, but usually the managers do not have understanding of the technical problems in the projects. In Germany and US they feel that they have management commitment to a satisfactory degree. The Germans feel that the management is interested in their problems and that they try to keep in good contact with them. This makes them not to feel too isolated from the mother organisation. According to the Total Quality Management approach management commitment and the long-term perspective are crucial for success of the change process [4].

In Denmark the project manager is more a member and co-ordinator of the team, while in US the project manager is on a higher level. Hofstede's Power Distance dimension explains this difference. The Power Distance Index (PDI) in Denmark is 18 in comparison with 40 for US and 35 for Germany. Low Power Distance means higher decentralisation and minimisation of inequality. [1] The manager is seen more as a co-ordinator and advisor without any direct power.

The authors point of view is that the Organisational Development (OD) approach, which is an umbrella term for a set of values and assumptions about organisations and the people within them, together with concepts and techniques, is useful for long-term organisation-wide change [4]. The OD approach cares about people and believes that people at all levels throughout an organisation are individually and collectively both drivers and engines of change. OD is a process by which behavioural knowledge and practices are used to help organisations achieve greater effectiveness, productivity, and improved product and service quality. The focus is on the process and on improving the organisation's ability to assess and to solve its own problems. It aims to improve the total system, the organisation and its parts in the context of the larger environment that impacts upon them [4]. The success of an OD approach lies in the capabilities of those who act as change agents or champions.

Decision Making

The decision making in the organisation is a delegated process, where all levels are encouraged to participate. In Germany and US the decision making is more hierarchical than in Denmark. It is usually the boss who takes the decision after consulting the software engineers. Because of higher power distance in US and Germany it is natural that the boss takes all decisions of importance, while in Denmark people on lower levels usually want to be part of the decision making process. Low Power Distance means higher decentralisation and minimisation of inequality [1]. The Boss is seen more as a co-ordinator without any direct power. Decisions in Denmark can be taken without the boss, who afterwards will simply be informed about the decision, while in US and Germany it is preferable that the boss is asked before the decision will be taken.

Meetings

The Danes seems to be more collaborative, while the Americans and the Germans are more hierarchical. In the Danish meetings everybody is given the opportunity to talk, while in the US and German meetings the manager asks for opinions of the participants but usually he/she dominates the discussion and it is also his/her opinion that will be taken most into consideration. In US they usually have one regular meeting per week, while in Denmark they have a lot of meetings.

Motivation

Most of the software developers are motivated by

- the work itself - the work has to be interesting and the objectives have to be clear
- recognition and acknowledgement of achievements, effort and knowledge both by managers and colleagues
- management commitment
- new technology and new tools
- ability to learn new things and develop as a professional
- the feeling of that the work makes sense
- influence on the daily work and how different problems are solved
- a good team spirit
- a feeling of being of same importance and equal part in a project

- to be respected as a colleague and professional (not as a German or an American)

The Americans seem to have a greater need for personal recognition and individual awards, while the Danes and the Germans are more concerned about team spirit and group belonging. This can be explained by Hofstede's Individualist - Collectivist Index. US score highest with 91 while Denmark and Germany are more similar scoring 74 and 67 respectively [1]. In individualist countries motivation should concentrate on the personal level, because people are supposed to take care of themselves and remain emotionally independent from the group. The dominant motivation is self-interest. In collective societies the concern is for the group. Individuals define their identity by relationships to others and group belonging.

Carrier path

There are four carrier paths in the organisation, namely:
- Line Manager
- Product Line
- Specialist
- Seniority

Most of the software engineers in Denmark have not thought very much about their carrier. They want to wake up in the morning and be happy to go to their work. They are in general proud of their work and do not worry much for the future. They are more concerned with quality of life. They work in order to live, they do not live in order to work. This has obviously to do with Hofstede’s uncertainty avoidance level, which is 23 for Denmark in comparison to 46 for US and 65 for Germany [1]. The Danes feel secure for their future and working in the organisation is a safe and not very competitive environment. The working environment is relaxing and people feel stressed usually only when they are out of schedule. In US the carrier is important but the size of the organisation is a barrier. There were plans from the beginning to enlarge the US organisation, but this has not happened. As a result there are not many possibilities for advancing in the hierarchy. In order to advance they understand that they have to be out-stationed to Denmark for some time. A more global aspect has been introduced recently by discussing the possibility of out-stationing on the interview stage for new employees.

The organisation mission

People do not seem to be aware of the organisation mission. They know that there is a mission, but it seems to be on a too high level. The word mission seems to be too abstract and they prefer to understand the mission more as a strategy or measurable objectives. In US they seem to be more aware of the mission. One of software engineer in US even had a small card with the overall organisation mission in his wallet. The organisational business goals and the importance of the projects are also invisible for the software engineers. They have difficulties to see how their project fit in the big picture. They cannot see the link between their daily work and the mission.

The global organisation

The organisation has geographically distributed software development in Denmark, Germany and US. The reason is mainly to be closer to the local markets, as well as dealing with difficulties of hiring resources locally. The opinion of the software developers is that the organisation is not technically really ready to be a fully global organisation in software development. The infrastructure is not set up for to be a global organisation. More investments in communication technology, on-line connections and integration of version control for both documentation, code and tools are required. An architecture that supports the global organisation has to be in place as well as a clear interface between the tasks so that the software engineers can work independently. Also cross-cultural training needs to take place stressing differences in the cultures for better understanding between the different sites and for a more effective organisation. Most of the software developers feel that they are working locally in the organisation. They do not feel that employees from other divisions are their colleagues. *'For me the corporation is the organisation'.*. Neither do they feel that they are very global. They have an understanding of that people from other cultures have a different mindset, another way of working and another way of being together and that they need to know the people they are going to work together with better in order not to feel *'we'* and *'they'*.

Conclusion

When building global teams it is important to chose the right people. Employees who have an open mind, who like new experiences and who have a cultural awareness. In a global project travelling should not be a constraint.. It is not always enough to pick up the phone and talk to each other's. People need to meet in order to get to know each other. It is an important part of promoting the sense of '*we*'.

'*In the beginning we did not really accept their behaviour, we were mourning about why do they behave like this. It took us about one year to understand them'*. This could have been avoided by some training in cross-cultural issues and also by an opportunity to getting to know each other's better. It is not only the direct questions and answers that are important, but also the informal discussions and information about details not seeming to be of big importance that give a background to work more professionally as a real team. In the beginning of a project it is important that the project team meets to discuss the requirements and to learn to know each other. The more often there is contact between the team members the better will the members of the team feel as one team.

The projects developed by virtual cross-cultural teams situated in different parts of the world are less effective. The current technology is a barrier for full effectiveness. The solutions are slower and there are misunderstandings. The different views of solving problems are an advantage that the organisation has to take into consideration. An essential base for utilising the cultural diversity is cultural awareness and an understanding of that people react differently. If people are willing to question their own way of working and to respect each other's the organisation will gain ultimate advantages.

In order to be a truly global organisation the individual work values must converge and be integrated into common set of values. On the same time the global organisation must adapt to the local cultures in which they operate in order to achieve high business performance [6]. The field-study proved the hypothesis [5] namely, that the organisation culture as well as the national culture has influence on the take-up and the awareness amongst the workforce of the software quality system

In order to achieve high software quality and high business performance process improvements have to take place continuously. The societal environment have been found to be important especially in transnational context [2, 3]. The iceberg metaphor is used to depict the contrasting aspects of organisational life [7]. The visible part of the societal iceberg emphasises the formal aspects of an organisation while the informal aspects of an organisation hide under water. The informal part is the greater part of the organisational iceberg and will act to help or hinder an organisational process of change. In this paper some of the informal aspects were analysed. Understanding these aspects enables IT managers at multinational organisations to operate more appropriately in countries other than their own.

References

[1] Hofstede Geert, 1994: Cultures and Organisations, Intercultural co-operation and its importance for survival, Software of the mind, *McGraw-Hill*, UK

[2] Ives B., Järvenpää S.: Applications of Global Information Technology: Key Issues for Management, *MIS Quarterly*, Vol 15 March 1991, pp. 33-49

[3] Keen P.W.: Planning Globally: Practical Strategies for Information Technology in the Transnational Firm in Palvia S., Palvia R., Zigli R. (Eds) *The Global Issues of Information Technology Management*, Harrisburg Pennsylvania: Idea Group Publishing, 1992, pp. 575-607

[4] Senior Barbara: *Organisational Change*, Financial Times Management, Pitman Publishing, 1997

[5] Siakas Kerstin V, Georgiadou Elli: Software Quality Management from a Cross Cultural Viewpoint, *Software Quality Journal*, Volume 8, No 2, October 1999 issue, pp 85-95

[6] Siakas Kerstin V., Georgidaou Elli: A New Typology of National and

Organisational Cultures to Facilitate Software Quality Management, *The fifth INternational conference on Software Process Improvement - Research into Education and training, INSPIRE 2000,* London 7-9 September, 2000

[7] Siakas Kerstin V, Georgidadou Elli: Process Improvement: The Societal Iceberg, *European Software Process Improvement Conference, EuroSPI '99,* Pori, Finland, 25-27.10.1999

The Perception of Quality Based on Different Cultural Factors and Value Systems

Gearoid O'Suilleabhain and Mary Jordan
DEIS, Ireland
Dr Richard Messnarz
ISCN LTD, Ireland
Dr Miklos Biro
Sztaki, Hungary
Ken Street
Telematica LTD, UK

Introduction

Many process improvement experts tend to have a narrow focus on their own area of research, leading to statements such as : Only Capability Maturity Model (CMM) will work; or, only Bootstrap will work; or, only SPICE (ISO 15504) is a comprehensive enough assessment framework; or, just skip all assessment methodologies and use goal based metrics, and so forth. [1]

Also, most of the improvement experts neglect the reality that human factors, skills, and social and teamwork competencies have significant influences on success. This is contrary to the belief of many process people who feel that processes can make the company independent of people.

In previous publications we promoted the experience of 30 experts from 11 different European countries and proposed a combined use of methodologies. These experts worked together for 3 years (1995 to 1998) in a European Leonardo da Vinci Project PICO (Process Improvement Combined apprOach). In particular, we focussed on the combination of business aspects with pragmatic improvement approaches as an essential requirement in order to make the right improvement decisions.

Furthermore, in the last 3 years, in a follow up project Bestregit, we analysed and identified factors that further impede your improvement efforts even if the goals, the business context, the methodology

and the plan for improvement are right. Human factors largely influence the behaviour of people, the effectiveness of teamwork, how the communication in the team works, and what can be achieved if the processes are right.
And one of the basic (probably funny) things is that different groups, cultures, and professions understand the term “Quality” differently.

Key Words: Software Process Improvement, Business Improvement, Strategies, Human Factors, Return on Investment, Innovation

Human factors

The Societal Iceberg

(K. Siakas, EuroSPI 1999, [20,31])

The iceberg metaphor shown in Figure 1 can be used to depict the contrasting aspects of organisational life.

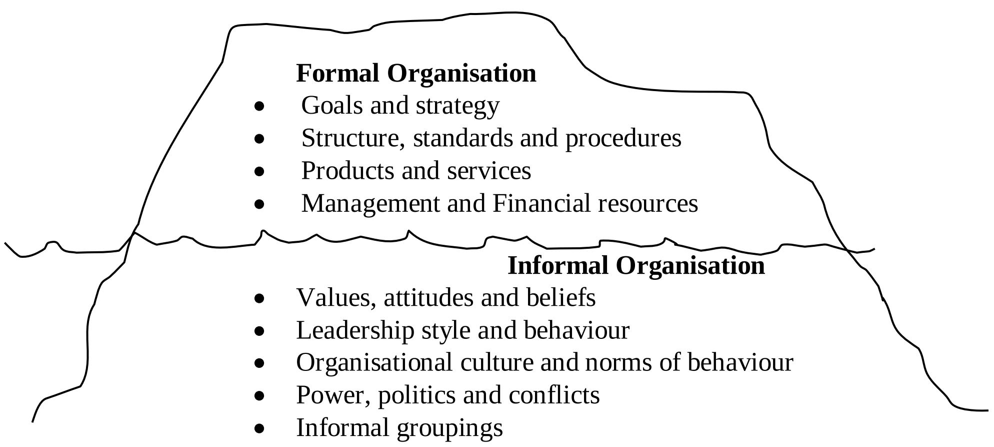

Formal Organisation
• Goals and strategy
• Structure, standards and procedures
• Products and services
• Management and Financial resources
Informal Organisation
• Values, attitudes and beliefs
• Leadership style and behaviour
• Organisational culture and norms of behaviour
• Power, politics and conflicts
• Informal groupings

Figure 1 :The organisational iceberg [20,31]

The visible part of the iceberg shows the formal aspects of an organisation, while the informal aspects of an organisation hide under water. The informal part is the greater part of the organisational iceberg and will act to help or hinder an organisational process of change. It often leads to resistance to the change process.

The view of organisations existing as systems of interrelated elements operating in multi-dimensional environments is becoming widely accepted. Both publications [22] and [23] report about Political, Economic, Technological and Socio-cultural factors that influence organisations in their structures, strategies, management process and means of operating, including technology and individuals.

The typical starting situation in a Bestregit project was a strong informal organisation with internal rules and personal habits in place, where goals are just in the heads of some people and the common sense of teamwork is not understood. There are some heroes fighting alone and believing that they are the best in their job. It takes a while (at least as far as completing the step of the goal analysis) for people to start to recognise common goals and interest, understand the management mission, and begin to acknowledge that larger goals can be achieved by forming a critical mass of teamwork.

This was the situation at all field test sites in Spain, Austria and Ireland at the beginning of the Bestregit project. This situation has also been observed and measured in a consulting project in IT industry at a major German firm. See below Figure 2 [1].

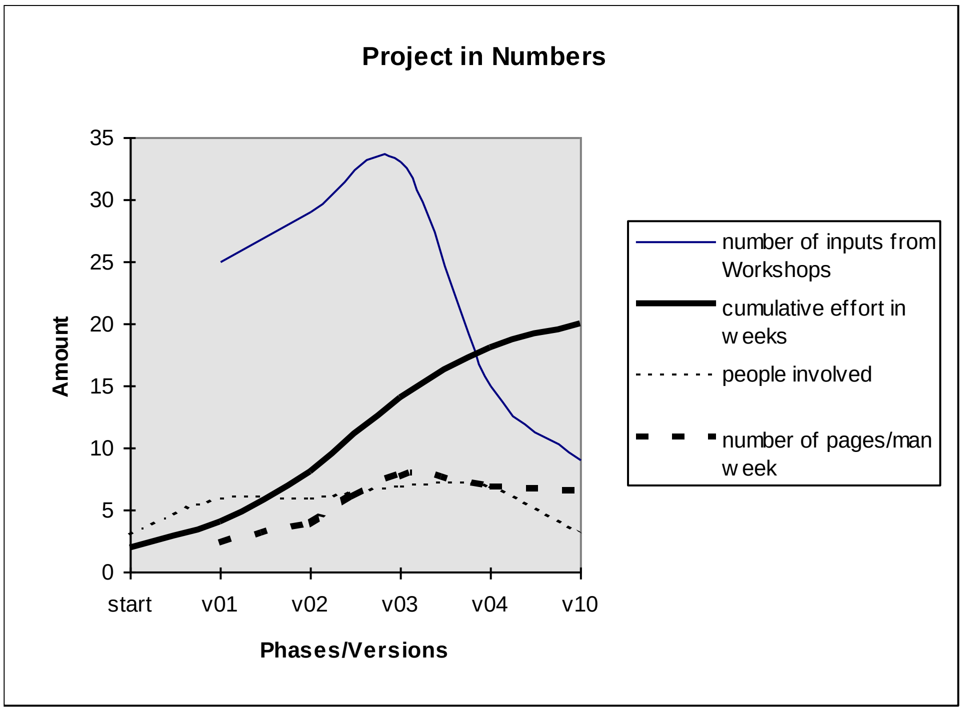


Figure 2: Modelling Project - Data review (PICO Book)

In a modeling project (see Figure 2), over a period of a year, data were collected to evaluate the performance and to see if lessons learned can be extracted. This resulted in two major people-oriented findings :

- Number of Inputs from the Workshop

If key people (from the informal part of the organisation) are left out from the workshop discussions, problems will be detected late in the process, usually when the staff complain that they cannot work with the model. Therefore it is a good sign if the input-curve increases in the first phases of the project and decreases towards the end of the project. The sooner you realise the requirements and problems the sooner you will get on the right track. The right selection of people to involve in the modeling is decisive.

- Cumulative Effort in Weeks and Number of Pages per Man Week

In the beginning the consulting and the field test team had a lot of discussions and interviews because it took time to understand the existing procedures and to deal with people ´s wishes and suggestions. So the modeling in the first two months was very slow. However, Once the goals, procedures, and missing practices were well understood and a team spirit was established, the modeling speed doubled (compare versions v02 and v03 in Figure 2). If you are on the right track, there must be a jump in your effort and productivity curve, otherwise you have not reached the "point of no return" when all staff and the modeling team start to work towards a shared vision.

In the Bestregit field test sites in Spain, Austria and Ireland this point of no return was achieved at a later stage than in the above described German project, due to additional cultural factors influencing the project:

- The Language Problem
- Culturally-determined Discrepancy in the Perceived Understanding
- The Cultural Diversity
- Reluctance to Change
- The Political Context
- Differing Background

The Language Problem

People speak different languages beside their native language

- A language based on their professional background (computer scientist, sociologist, lawyers, etc. speak very different languages and use terms that are specific to their profession).
- A language based on their position (directors, managers, staff, and administrative personnel have different languages).

Position Language. It is a fact that directors, managers, and staff speak different languages and may have different viewpoints on the same situation. This has been observed in a field study in the US involving above 70 companies. An interview of business managers (directors) and project managers on the same topics illustrated -

- They only agreed on one third of the topics and they disagreed on the remaining two thirds.
- They agreed on improvement potentials in areas such as unrealistic project planning, inability to detect problems early, insufficient number of checkpoints.
- They disagreed on improvement potentials in areas such as customer management, progress tracking, staffing problems, technical complexities, project scopes.

More details can be found in [1].

Professional Language. Bestregit was not only a multi-cultural project, it was also an inter-disciplinary one. The participating people had largely differing backgrounds including computer science, history, languages, business, sociology, and law. Thus, when we started with an improvement guideline version 1.0 written in a computer science jargon many of the users failed to understand the document fully.

This situation was solved in two ways:

First a ***common notation*** was greed to express models and goals in a simple way. The notation had to be simple enough to be understood by non-computer background people. This formal language was then used in all further meetings by all partners as a basis to discuss their results with the others.

Secondly the guideline had to be refined once again to make more definitions and explain the improvement steps in a non-computer-scientist specific jargon. Thus a success criteria was the use of an ***inter-disciplinary language*** that allows common understanding of principles and steps across professions such as computer science, history, languages, business, sociology, and law.

The misunderstandings caused by different professional languages at the beginning of the project can be illustrated by the following facts:

- In the first kick off meeting the general managers stopped the presentation of the computer science people after two slides (of 40) because things were completely misunderstood.
- The number of meetings were nearly 5-6 a year at the beginning, re-discussing the same issues again and again, whereas after reaching the “point – of – no – return” this was reduced by half for the rest of the project.

Only after a common notation was used and the language had been refined to be understandable by all the “jump-in-the-effort-curve” was achieved. (see Figure 2).

Discrepancy in the Perceived Understanding

In [40] in a project “OSIRIS” a practical example of the Johari window is shown and how discrepancy in the perceived understanding was measured.
The Johari window is a tool originally designed for conceptually distinguishing between 4 different possible states with regard to knowledge of oneself, these states are shown in the diagram below:

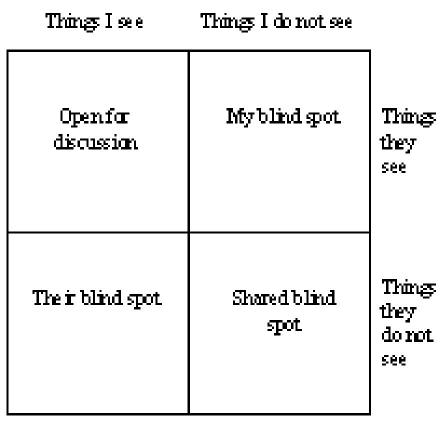


Figure 3: The Johari Window

The same principle can be applied to measure 4 different possible states with regard to knowledge of one’s culture, and the understanding of other cultures in the project team.

Respondents to the OSIRIS survey were asked to rate, on a scale from 1 to 5, the discrepancy between the understanding cultures in the ISIS project have of themselves, and the understanding others have of them. The questions asked were based on the Johari Window.
The table below shows the perceived discrepancies between each culture’s understanding of itself and the understanding other cultures have of it. Note that the ratings for each culture are by the <u>other</u> two cultures not by own culture.

	%	*Mean*	*Std. Deviation*
Discrepancy between German respondent's own understanding of themselves and that of respondent	15.3	3.7500	.5000
Discrepancy between Irish respondents' own understanding of themselves and that of respondent	46.2	2.3333	.7785
Discrepancy between Greece's respondents' own understanding of themselves and that of respondent	38.5	2.4000	.8433

Figure 4: Measured Discrepancies in the Perceived Understanding

The bigger the discrepancy the smaller the incidence of perceived shared understanding and, implicitly then, the fewer issues “open for discussion”. It is interesting that Germany was the only culture with an above-average mean score which may imply that other cultures more often than not find their own understanding of German culture to be at odds with that of the Germans' own and that "blind spots" , which can hinder effective interaction, may be greater than the Germans themselves believe.

Nearly the same situation appeared in the Bestregit project. The project leader and the consulting partner who developed the improvement guideline were native Austrian. In technical science issues, and perhaps also due to the same native language, Austrians are similar in this respect to the Germans.

It happend during the project that some results (e.g. the interactive training course) had to be re-agreed a number of times although the Austrian group felt that clear definitions have been agreed already.

The Cultural Diversity

Layers and dimensions of national, organizational, and other group cultures have been identified by thorough scientific research. Layers of culture from the most superficial to the deepest are briefly summarised here:

- *Symbols*: words, gestures, pictures, objects that carry a particular meaning which is only recognized by those who share the culture.
- *Heroes*: persons, alive or dead, real or imaginary, who possess characteristics which are highly prized in a culture, and who thus serve as models for behavior.
- *Rituals*: collective activities technically superfluous in reaching desired ends, however socially essential.
- *Values*: broad tendencies to prefer certain states of affairs over others.

The first 3 layers are visible to an outside observer; their cultural meaning, however is invisible. The 4th layer, values, are acquired so early in our lives that they remain unconscious. Therefore they cannot be discussed, nor can they be directly observed by outsiders.

Geert Hofstede surveyed 116,000 IBM employees across 40 countries and identified four important dimensions of national value systems. Hofstede's four dimensions [13,23] are:

Power Distance :Power Distance Index (PDI) indicates the extent to which a society accepts the fact that power in institutions and organisations is distributed unequally among individuals. In small PDI countries subordinates and superiors consider each other as existentially equal and decentralisation is popular, while large PDI countries subscribe to authority of bosses and centralisation.

*Collectivism / Individualism:*Individualism indicates the extent to which a society is a loose social framework in which people are supposed to take care only of themselves and their immediate families. Collectivism is a tight social framework in which people distinguish between in-groups and out-groups and expect their in-group to look after them. In individualist countries people are supposed to take care of themselves and remain emotionally independent from the group. The dominant motivation is self-interest. In collective societies the concern is for the group. Individuals define their identity by relationships to others and group belonging.

Femininity / Masculinity: Masculinity indicates the extent to which the dominant values in a society tend toward assertiveness and the acquisition of things. In masculine cultures importance is placed on assertiveness, competitiveness and materialism in the form of earnings and advancement, promotions and big bonuses. Femininity indicates the concern for people and the quality of life. In feminine cultures the concern is for quality of relationships and the work of life, nurturing and social well being.

Uncertainty Avoidance: Uncertainty Avoidance indicates the extent to which a society feels threatened by ambiguous situations and tries to avoid them by providing rules, believing in absolute truths, and refusing to tolerate deviance. In weak uncertainty avoidance countries anxiety levels are relatively low. Aggression and emotions are not supposed to be shown and people seem to be quiet, easy-going, indolent, controlled and lazy while in high uncertainty countries people seems to be busy, fidgety, emotional, aggressive and active.

All the four dimensions are a continuum between two extremes and only very few national cultures, if any, are wholly at one or the other extreme.

According to Calori [33] four clusters emerge from Hofstede's European results.

- A Germanic Group (high masculinity and low power distance)
- A Scandinavian Group (high individualism, low masculinity, and low power distance)
- An Anglo-Saxon Group (high individualism and masculinity, and low power distance and uncertainty avoidance)
- A Latin Group (high uncertainty avoidance and high power distance)

It is interesting that each of the 3 field test partners belongs to a different cluster (Austria to the Germanic Group; Ireland to the Anglo-Saxon Group, Spain to the Latin group).

A subjective impression from the Bestregit project suggests that in Europe we already exchange cultures to such an extent that people who have been involved in mobility projects recognise and display a mixture of cultural indicators.

e.g. The Irish expert organising the field test in Ireland was native Irish but had lived in Poland a long time.
e.g. The Spanish expert organising the field test in Spain was native Spanish but had lived in England for more than one year.

And so forth.

Of course, we learn from other cultures, and especially good and desirable attitudes are likely to be copied.

Individualist or collectivist value systems fundamentally influence the perception and usefulness of group decision systems and teamwork infratstructures. Hofstede [13] refered to a management researcher from the U.S., Christopher Earley who performed an enlightening laboratory experiment on a group of 48 management trainees from southern China and 48 matched management trainees from the U.S. Half of the participants in either country were given group tasks, the other half individual tasks. Also, half of the participants in either country, both from the group task and from the individual task subsets were asked to mark each completed item with their names, the other half turned them in anonymously. “The Chinese collectivist partcipants performed best when operating with a group goal and anonymously. They performed worst when operating individually and with their name marked on the items produced. The American individualist participants performed best when operating individually and with their name marked, and abysmally low when operating as a group and anonymously.”

This means that different continents will have different requirements from the same methodology, and Bestregit would have to be promoted differently in China than in Europe or the US. However, the higher the acceptance of group decisions the more effective a Bestregit approach would become.

Reluctance to Change – Heroic Defense
The study from Hofstede was performed within IBM sites and thus mainly addressed people in the software and electronics sector within one and the same company. However, the basis for the Bestregit project and field tests was general (pubic) and non-profit service organisations who are working on behalf of their states and the European Commission. So the environment is not really comparable and the Hofstede results not directly applicable.

These service organisations are usually integrated into existing research, educational, industry, or state-service networks and have a quite hierarchical structure. Decisions must be agreed with a director, presented and agreed on a board (sometimes with representatives from social partners, local industry, the government) and must be planned with the major funding sources.
Thus managers in these "innovation transfer organisations" have to play roles that are sometimes at odds with each other –

- They must show high individualism to find new innovative methods and ways*: The Innovation Agent's View*.
- They on the other hand should be controlled by all the above mentioned parties and individualism is required to be low: *The Board View*.

Mostly the board consists of state officers and senior managers of local industry (with a high uncertainty avoidance), whereas the innovation agents are contracted consultants with high individualism and low uncertainty avoidance.

When Bestregit was presented for the first time, the board representatives (with a high unertainty avoidance) did not see an urgency for change. A typical view of people with high uncertainty avoidance was: Why change the world, if everything has been stabilised so nicely?

The experience in the Bestregit project shows that these people with a high uncertainty avoidance run through a set of self-experience stages:

- At the beginning they let Bestregit start as a useful project but remained skeptical.
- After the goal analysis they start to understand what is going on but still smile about the goals.
- After the teamwork analysis they see how the work force restructures and that the models create a better image to outsiders (as a quality organisation), attracting further funding. At this stage they get interested and ask questions and want to understand the details.
- In the experimentation phase it is the first time that they see measurable results and start to react positively, and allowing their immutable, secure world to change.

From a cultural perspective (taking Hofstede results and these additional factors into account) the following impacts became visible in the Bestregit project.

- In Austria (by the board), in Ireland and in Spain (through changes of the managers) in the first 9 months of the project it was given from one manager to the other like a hot iron, holding it too long would burn her/his fingers.
- Once a manager with high uncertainty avoidance was found, who still supported the requirements of change, movement occurred.
- Austria overdid the modeling and produced more results than required.
- Ireland caught up quickly with much individualism, constantly entering many new requirements.
- Spain caught up quickly as well, but always applied a certain freedom in the implementation (e.g. combining notations to describe all materials in fewer pages).

So the impression is that what Hofstede published for IBM might show different results if it is applied for a different industry sector.

Process Improvement Driven by Business

In the U.S. the use of assessment methods became inevitable in the mid 80's. A large part of the software systems funded by the DoD (Department of Defense) were actually never used because their quality was not sufficiently checked and error behaviour could not be predicted making it a hazard to use such systems in critical military situations. In Europe (except the UK) the military industrial sector was not the most important business. It was banking, insurance, manufacturers (cars), and so forth.
So it was much easier in the U.S. to motivate companies to use process improvement approaches

because the larger funds ran through DoD and organisations were eager to search for this funding.
In Europe this was much more difficult. The military sector is not the strongest driving force, it is rather market demands (to show ISO 9001 compliance for achieving customer confidence), supplier relationships (to satisfy the customer and get further contracts), and ambitions to achieve a competitive structure for the future organisation.
The major question in Europe is (and was), if there is no "must" through the military sector, what can motivate business managers to provide necessary funds and support for process improvement initiatives.

People as an Asset of the Company
While Western Europe was surrounded by two worlds, an extreme communist view in Eastern Europe and a strong capitalist view in the U.S., most European nations went through a political process that created a kind of social capitalism in the last century.
In Italy, for instance, the communist party is very strong and all other parties had to create an alliance to keep the communist parties out of power as long as there was the cold war. In France, for instance, there is a history over hundreds of years and revolutions in which the principles of "liberte", "fraternite", and "egalite" were kept alive and led to well defined social rights and labour laws which protect employees and their rights.
Also in Germany, big electronics firms have founders who were people with a social conscience who created a synergy between their company and the social rights and social support of people.
So European employees have much more rights and protection than in the U.S., which created a much more people oriented approach, and motivation becomes a key asset.

It is not enough in Europe to just define processes, assess them, and, if they are not properly executed, to set staffing consequences or exchange people. This is the reason why many European organisations try to establish processes with a team-work culture that allow people to identify themselves with the process and to actively work with others in a highly motivating environment.

Goals as a Translation of Different Viewpoints
As a consequence from the above descriptions, a huge task in a people-centred environment is to reach a critical mass of people that follow a common vision and are motivated to achieve the business manager's goals in process improvement.

A major problem is the fact that different languages are spoken, and it seems hard to find a consistent architecture of goals that links the objectives of the different groups in such a way that success and the achievement of goals becomes traceable.
A goal tree then becomes a translator of process improvement understanding across different groups within and outside an organisation.

Bestregit took all these aspects into account and build them into the methodology, through a continuous interaction between the improvement guideline developer and field test partners from different cultural environments.

Differing Background and Skills

In Europe the education systems are very different. For instance, a software engineer from UK has different qualifications than one from Germany, although in a global market place they operate in the same position in an organisation.
Also people collect different experiences over the time, which upgrade their personal knowledge, but make them different from others.
So an organisation consists of many different individual brains (experiences) and the knowledge gathered in them. The processes are then a means to focus that knowledge for the success of the firm.

A way to focus and direct this knowledge is to establish a skill profiling and learning system which helps engineers from the workplace to get their internal advise on a personal learning route, and the system ensures that (although offered on a personal level) a common focus is achieved.

Here we refer to the EU project CREDIT which developed a generic (configurable) environment which can be adapted to different skill sets, learning routes, and offers guidance through an Intranet system.

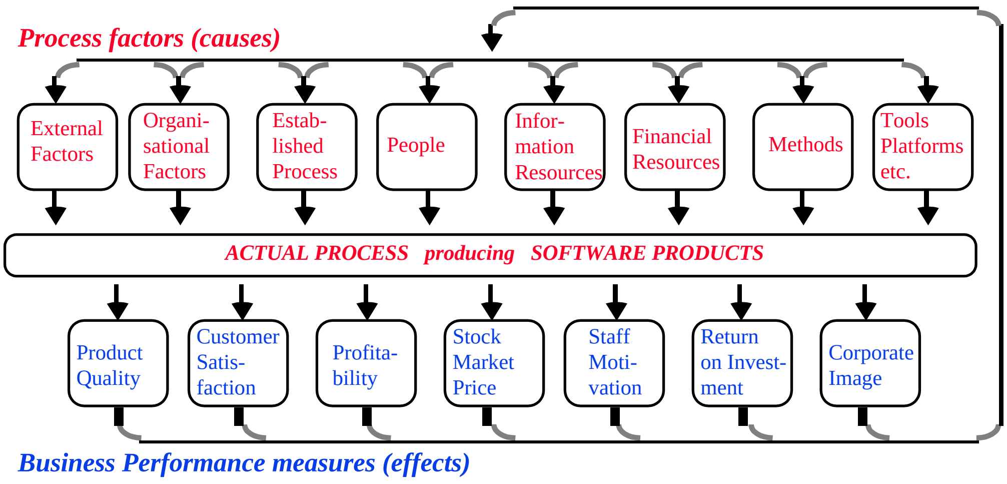


Figure 5: The CREDIT System Architecture

What does quality mean?

Different Viewpoints on Quality from Different Positions

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

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

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

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

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

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

"**Manager's view.** A manager may be more interested in the overall quality rather than in a specific quality characteristic, and for this reason will need to assign weights, reflecting business requirements, to the individual characteristics.
The manager may also need to balance the quality improvement with management criteria such as schedule delay or cost overrun, because he wishes to optimise quality within limited cost, human resources and time-frame."

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

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

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

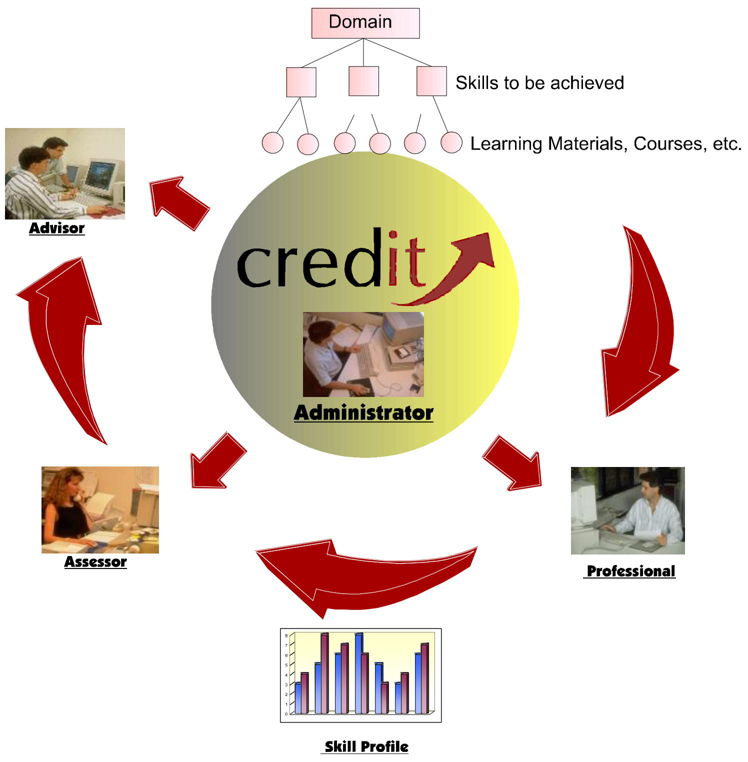


Figure 1 : A Process – Business Feedback Loop

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

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

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

Conclusion

The presentation of these various models and theories of culture and its influence is a way of empathising the ubiquitous and often powerful force of cultural-determined differences. Culturally-determined differences can, of course, be both a help and a hindrance in any kind of cross-cultural collaboration and efforts at modelling across cultures. If the relative strengths of each culture can be appreciated then they can be drawn from and used to strengthen cross-cultural collaborations. As a result, however, of what Susan Schneider has called the "convergence myths" of international business, that the world is getting smaller anyway (theories of a global information society belong to this myth) and that business is business no matter where you are, the cross-cultural businessperson may choose to believe that cultural differences are either non-existent or of no importance. Such a belief can be the ruination of many efforts at co-operation across cultures.

It is not that culture is the most important factor in such undertakings merely that because it is so commonly overlooked it may well be one of the most common reasons for the failure of such undertakings to enjoy full success. In this relatively short presentation we hope we have touched upon some of the issues involved in the often unrecognised influence of culture upon the field of process analysis and specifically upon the perception of quality in this field. Cultural factors such as those we have grouped under the headings of Language; Culturally-determined discrepancies in perceived understanding; Cultural profiles based on Hofstede's studies and other cross-cultural studies; National identities and national systems of education and the organisation of work and business cannot be ignored. An understanding of how these factors may influence a process improvement experiment, an organisation's goal or an individual's perception of quality is an important a weapon in the arsenal of those attempting to facilitate process improvement as the knowledge of the many various traditional models and methodologies for achieving this aim. Indeed it may be time to modify many of these models to take into account the influence of cultural factors.

References

[1] Messnarz R. Tully C., Better Software Practice for Business Benefit – Principles and Experience, IEEE Computer Society Press, Brussels, Washington, Tokio, 1999, ISBN : 0-7695-0049-8.

[2] Alavi M., Carlson P.: A review of MIS Research and Disciplinary Development, Journal of Management Information Systems, 8 (4) pp. 45-62
[3] Mumford E., Hirschheim R., Fitzgerald G., Wood-Harper A.T (Ed.): Research Methods in Information Systems, Elsevier Science Publishers
[4] Galliers Robert Ed.: Information Systems Research, Issues, Methods and Practical Guidelines, Blackwell Scientific Publications
[5] Von Hellens L.A: Information systems quality versus software quality. A discussion from a managerial, an organisational and an engineering viewpoint. Information and Software Technology, 39, 1997, pp. 801-808
[6] Ives B., Järvenpää S.: Applications of Global Information Technology: Key Issues for Management, MIS Quarterly, Vol 15 March 1991, pp. 33-49
[7] Keen P.W.: Planning Globally: Practical Strategies for Information Technology in the Transnational Firm in Palvia S., Palvia R., Zigli R. (Eds) The Global Issues of Information Technology Management, Harrisburg Pennsylvania: Idea Group Publishing, 1992, pp. 575-607
[8] Tractinsky Noam, Järvenpää: Information Systems Design Decision in a Global versus Domestic Context, MIS Quarterly/ December 1995, pp. 507-529
[9] Hunter M. Gordon, Beck John E.: A cross-cultural comparison of "excellent' system analysts, Information Systems Journal, 1996, 6, pp. 261-281

[10] Couger, J.D. Adelsberger H. (1988): Comparing motivation of programmers and analysts in different socio/political environments: Austria compared to the United States, Computer Personnel, 11(4), pp. 13-17
[11] Couger J.D., Borovitz I., Zviran M.: Comparison of motivating environments for programmer/analysts and programmers in the US, Israel and Singapore In Sprague, R/H. Jr (eds) Proceedings of the 22nd annual Hawaii International Conference on Systems Sciences, IEE Computer Society Press, Washington, DC, 1989, pp. 316-323
[12] Ein-Dor P., Segev E., (1993): The effect of national culture on IS: implications for international information systems. Journal of Global Information Management, 1, pp. 33-44
[13] Hofstede Geert: Cultures and Organisations, Intercultural co-operation and its importance for survival, Software of the mind, McGraw-Hill UK, 1994
[14] Eriksson Inger, Törn Aimo: Introduction to IST Special Issue on Information System Quality, Information and Software Technology, 39, 1997, pp.797-799
[15] Kendall Julie E., Avison David: Emancipatory research themes in information systems development: Human, organisational and social aspects in Human, Organisational. And Social Dimensions of Information Systems development (A-24) edited by Avison D., Kendall J.E., DeGross J.I., Elseviers Science Publishers, IFIP, North-Holland, 1993, pp. 1-11
[16] Curtis Bill, Heflleey William E, Miller Sally. Overview of the People Capability Maturity Model, CMU/SEI_95-MM-01, 1995
[17] Jack, Rickard. Personal Issues in SoftwareCost Estimation, 5th Software Quality Conference, 9-11 July, Dundee, 1997
[18] Hodge B.J., Anthony W.P. Organisational Theory, Allyn and Bacon, Boston, 1988
[19] French W.L., Bell C.H.: Organisation Development: Behavioural Science Interventions for Organisation Improvement, Englewood Cliffs, NJ, Prentice- Hall International, 1990
[20] Johnson G., Scholes K.: Exploring corporate Strategy, 3rd edn, Hemel Hempstead, Prentice Hall, 1993
[22] Goodman M.,: Creative Management, Hemel Hempstead, Prentice Hall , 1995
[23] Senior Barbara: Organisational Change, Financial Times Management, Pitman Publishing, 1997
[24] Benjamin Robert: Managing Information Technology enabled Change In Human, Organisational. And Social Dimensions of Information Systems development (A-24) edited by Avison D., Kendall J.E., DeGross J.I., Elseviers Science Publishers, IFIP, North-Holland, 1993 pp. 381-398
[25] Cummings T.G., Worley C.G. (1993), Organisation Development and Change (5th edn) St Paul, MN, West Publishing Company
[26] Geletkanycz Marta A.: The salience of 'Culture's consequences': The Effect of Cultural Values on Top Executive Commitment to the Status Quo Strategic Management Journal, 1997, Vol.18:18, pp. 615-634
[27] Orlikowski Wand J., Baroudi Jack J. Studying Information Technology in Organisations: Research Approaches and Assumptions, Information Systems Research 1991,2:1, 1-28
[28] Land Frank ,The Information Systems Domain in Information Systems Research, Issues, Methods and Practical Guidelines ed. Galliers Robert
[29] Ralston David A., Holt David H., Terpstra Robert H., Kai-Cheng You The Impact of National Culture and Economic Ideology on Managerial Work Values: A Study of the United States, Russia, Japan and China, Journal of International Business Studies, 1st Quart. 1997, pp.177-205
[30] Mohamed Walaa Eldeen and Siakas Kerstin V. (1995): Assessing Software Quality Management Maturity (SQMM): A new model incorporating technical as well as cultural factors, the 3rd International Conference on Software Quality Management SQM95, 3-5 April, 95, Seville, pp. 325-336
[31] Siakas Kerstin V. The Effect of Cultural Factors on the Implementation of Software Quality Management Systems, 5th Software Quality Conference, Dundee, 9-10 July, -96
[32] Newman Karen I., Nollen Stanley D: Culture and congruence: The Fit between Management Practices and National Culture, Journal of International Business Studies, 4th Quarter 1996, pp.753-777
[33] Calori, Roland. A European management model : beyond diversity. New York ; London : Prentice Hall, 1994
[34] Daft, Richard. Management. Fort Worth, TX : Dryden Press, 1997
[35] Loader, Brian. Cyberspace divide. London ; New York : Routledge, 1998
[36] Lyon, David. The information society : issues and illusions. Oxford: Polity, 1988
[37] Miles, Ian. Information technology and information society. London: ESRC, 1988
[38] Palloff, Rena. Building learning communities in cyberspace. San Franciso: Jossey-Bass, 1999
[39] Schneider, Susan. Managing across cultures. London: Prentice Hall, 1997.

[40] Gearóid Ó Súilleabháin, MScEcon., DEIS, Cork Insitute of Technology, OSIRIS Project - Cross-cultural Collaboration: the experiences of the LEONARDO ISIS project, May 2000

Session 3 - SPI Implementation

Session Chair:

Yingxu Wang,

IVF

Avoiding Failure in SPI Initiation

Pouya Pourkomeylian
AstraZeneca R&D Mölndal, S-431 83 Mölndal, Sweden
Tel: +46 31 776 17 96, Fax +46 31 776 37 30
Viktoria Institute, Box 620, 495 30 Göteborg, Sweden
Pouya.pourkomeylian@astrazeneca.com

Abstract

Software Process Improvement (SPI) is a recognised systematic approach for improving the capability of software organisations. Such initiatives have however proven to meet with a number of difficulties such as: scaling the SPI initiatives, setting realistic goals, the complexity of organisational changes, and the organisational culture. For organisations with no earlier experience with SPI, the first initiative might therefore run the risk of being the last.
This paper shows the results of a case study in which the first SPI initiative in an organisation was analysed according to two frameworks: first a framework that maps the characteristics features of SPI and second a framework that describes SPI as a knowledge creation process. On the basis of our findings we argue that the first SPI initiative: 1) should be a learning process focusing on learning SPI practice, 2) should satisfy organisational goals rather than routinely follow a normative model for reaching any maturity level, and 3) should be organised as a project aiming to improve a few software processes based on practitioners' ideas. These insights are compared and contrasted with the established knowledge on how to initiate SPI activities.
Keywords: SPI Initiation, Knowledge Creation.

1. Introduction

Software Process Improvement (SPI) is a systematic approach to improve software processes in organisations. This approach was developed by the Software Engineering Institute (SEI) inspired by the work of Watts Humphrey [1]. The basic idea of SPI is to focus on software processes as social institutions with a complex interplay of people, methods, tools, and products [2], figure 1.
SPI initiatives start with an assessment to identify organisations' current software process problems. The improvement activities should then be planned and performed on the basis of the assessment's findings and other goals of the organisation. The improved new software processes should then be institutionalised in the whole organisation to become a part of the practitioners' daily work. Many organisations have been inspired by the concept of SPI and started SPI initiatives. However achieving success with SPI has been shown to be a difficult challenge for many organisations. Many organisations do not succeed in performing their improvement activities, others have difficulties with implementation of new processes in the organisation [3]. Different factors such as scaling the SPI initiative, setting realistic goals, the complexity of organisational changes, and the organisational

culture have made it difficult to achieve success in SPI initiatives [4], [5], [6], and [7]. For an organisation with no earlier experience in SPI (a novice organisation), the first initiative might therefore run the risk of being the last. Therefore, it should be crucial for such an organisation to find answers to some key questions before starting an SPI initiative: What are the most characteristic features of the first SPI initiative? How should a novice organisation organise, plan, and conduct an SPI initiative? Does a novice organisation have a fair chance of succeeding in its first SPI initiative?

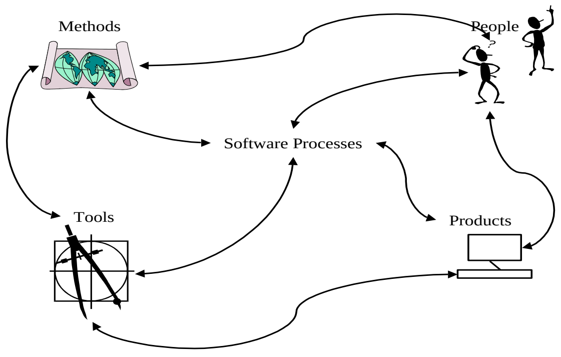


Figure 1: A complex interplay of people, methods, tools, and products.

According to Aaen *et al.* [8], organisations that start SPI efforts should find inspiration and guidance in the literature. They argue that these organisations should avoid the pitfalls that have led to failure in other organisations and should learn from successful initiatives that have bearing on their own situation. However, following this advice is not easy: the SPI literature is extensive and is growing and there are no authoritative sources outlining the underlying rationale of the SPI [8]. A large body of knowledge about SPI has become available during the last years, including specific models [9], [10], concepts to support practical use of the models [11], [12], experience reports [4], [7], and critical evaluations [13]. A survey of SPI literature and a MAP of the key ideas in SPI are presented by Aaen *et al* [2]. They provide a conceptual MAP of software process improvement which describes three fundamental concepts of SPI approach including nine ideas, i.e. the *management* of SPI activities, the *approach* taken to guide the SPI initiatives, and the *perspective* used to focus attention on the SPI goals. The MAP also addresses the following questions: 1) What are the characteristic features of SPI initiatives? How do SPI initiatives compare with other improvement approaches? 3) What are the key benefits and risks related to SPI initiatives?
Using the MAP, this study tries to analyse one SPI project done as the first SPI initiative in a novice organisation. The main question in this paper is: Which were the crucial features in successfully initiating this SPI project? It is hoped that this analysis will help other novice organisations to find ways to increase their chances of success and minimise the risks of failure.

2. Research Approach

This study uses a case study approach to analyse and map an SPI project performed as the first initiative in a novice organisation. Case studies are particularly useful for an understanding of some special subject in great depth and where one can identify cases rich in information, rich in the sense that a great deal can be learned from a few examples of the phenomenon in question [14]. This approach was chosen for this study because of its strength in focusing on the case and its ability to provide techniques to structure the research process and its findings.
The SPI initiative was carried out from April 1999 to May 2000 and aimed to improve the software organisation's software processes. This paper does not include the activities related to the implementation of new processes within the software organisation. The author has been the driving force behind the SPI initiative and has actively participated in activities to initiate, organise, plan, and conduct the SPI initiative during the one-year period. In this study the author reflects on the SPI initiative and tries to make conclusion about, lessons useful for understanding the field of SPI and

support the practice of the first SPI initiative in novice organisations.
The next section discusses the knowledge creation and learning concepts and SPI's characteristic features. Section 4 presents the case. Section 5 presents the results of the SPI initiative analysed and discusses the findings according to the research question stated above and section 6 concludes the paper by presenting the lessons learned and pointing out areas for further research.

3. A Theoretical Framework

This section presents a brief description of the two frameworks used to analyse the SPI initiative.

3.1 Knowledge Creation and Learning

Nonaka and Takeuchi [15] use two dimensions of knowledge creation to explain the process of organisational knowledge creation: 1) *the ontological* and 2) *the epistemological*. The *ontological* dimension focuses on individual knowledge creation. The organisation supports creative individuals or provides a context for them to create knowledge. Organisational knowledge creation is therefore understood as a process that "organisationally" amplifies the knowledge created by individuals and crystallises it as a part of the knowledge network of the organisation. This process takes place within an expanding "community of interaction", which crosses intra- and inter-organisational levels and boundaries. For the *epistemological* dimension Nonaka and Takeuchi [15] draw on Michael Polanyi's [16] distinction between explicit knowledge and tacit knowledge. Explicit knowledge refers to knowledge that is transmittable in formal, systematic languages. It can be articulated in formal languages including grammatical statements, mathematical expressions, specifications, manuals and so forth. It can be transmitted across individuals formally and easily. Tacit knowledge is personal and context-specific and therefore difficult to formalise and communicate. It is personal knowledge embedded in individual experience and involves intangible factors such as personal belief, perspective, and the value system. Tacit knowledge is difficult to communicate and share within the organisation and must thus be converted into words or numbers that everyone can understand.
Nonaka and Konno [17] describe two dimensions to tacit knowledge. The first is the *technical* dimension, which encompasses the kind of informal personal skills or crafts often referred to as "know-how". The second dimension is the *cognitive* dimension, which consists of beliefs, values, ideals and mental models that are deeply ingrained in us and that we often take for granted. They argue further that this cognitive dimension of tacit knowledge shapes the way we perceive the world and creates our understanding of it. According to Nonaka and Takeuchi [15] the organisational knowledge is created during the time the "*conversion*" takes place, i.e. from tacit to explicit and back again into tacit. The interaction between these two forms of knowledge is the key dynamic of the knowledge creation in the organisation.

3.1.1 Knowledge Conversion

Knowledge conversion is a "social" process between individuals and not confined within an individual. Assuming that knowledge is created through the interaction between tacit and explicit knowledge, four different modes of knowledge conversion are possible (Figure 2). The contents of the knowledge created by each mode of knowledge conversion is naturally difficult, which create different contents of knowledge [15]:

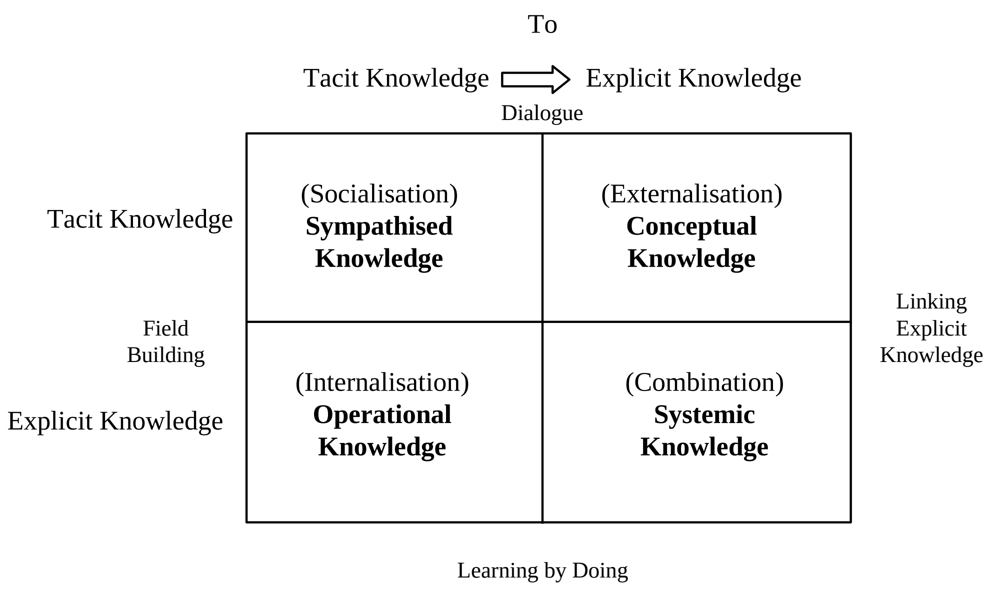


Figure 2: Contents of knowledge created by the four modes [15].

1. From tacit knowledge to tacit knowledge (*socialisation* which creates *sympathised* knowledge). The socialisation mode usually starts with building a "field" of interaction. This field facilitates the sharing of members' experiences and mental models. Socialisation involves the sharing of tacit knowledge between individuals.
2. From tacit knowledge to explicit knowledge (*externalisation* which creates *conceptual* knowledge). The externalisation mode is triggered by meaningful "dialogue or collective reflection," in which using appropriate metaphor or analogy helps team members to articulate hidden tacit knowledge that is otherwise difficult to communicate. Externalisation requires the expression of tacit knowledge and its translation into comprehensible forms that can be understood by others.
3. From explicit knowledge to explicit knowledge (*combination* which creates *systematic* knowledge). The combination mode is triggered by "*networking*" newly created knowledge and existing knowledge from other groups in the organisation, thereby crystallising them into a new product or service.
4. From explicit knowledge to tacit knowledge (*internalisation* which creates *operational* knowledge). "*Learning by doing*" triggers internalisation. The internalisation of newly created knowledge is the conversion of explicit knowledge into the organisation's tacit knowledge. In practice, internalisation relies on two dimensions. First, explicit knowledge must be embodied in action and practice. Second, there is a process of embodying the explicit knowledge by using simulations or experiments to trigger learning by doing processes.

3.2 The most characteristic features of SPI

To have a better chance of succeeding with SPI efforts an organisation must understand the most important key ideas in SPI. Understanding SPI includes knowing: 1) what elements affect the SPI initiative, 2) how to organise, plan, and take action to make SPI happen and 3) how to institutionalise the new processes within the organisation.
Aaen *et al.* [2] present a map structured mainly on the basis of theoretical insight in the SPI literature and experience in practising SPI in close collaboration with software organisations and describe the characteristic features of SPI initiatives. According to these authors SPI is based on a number of ideas that offer specific answers to specific concerns. SPI has three fundamental concerns: the *management* of SPI, the *approach* taken to guide the SPI initiatives and the *perspective* used to focus attention on the SPI goals. Table 1 contains a survey of the MAP described by Aaen *et al.* [8] and provides an overview of the key ideas involved in SPI. The table gives information about alternatives,

opportunities, and risks related to SPI initiatives.

Concern	Idea	Aspiration	Pitfalls
Management of SPI	Organisation	Dedicated and adapted effort	Inadequate resources, emphasis and co-ordination
	Plan	Plan goals, activities, responsibilities and co-ordination	Loss of motivation. Diversity or deadlock
	Feedback	Measure and assess benefits	Opportunism, and loss of relevance
Approach to SPI	Evolution	Experiential learning and stepwise improvement	Wearing and inertia
	Norm	Seek dedication and legitimacy	Hastiness and fundamentalism
	Commitment	Ensure dedication and legitimacy	Goal deflection and gold plating
Perspective in SPI	Process	Integrate people, management and technology	Disinterested customers
	Competence	Empowerment through competence building	Turf guarding
	Context	Establish sustainable effort	Machine bureaucracy

Table 1 : The SPI MAP with aspiration and pitfalls [8], p 2.

Based on their concept the management of SPI initiatives builds on three ideas: 1) the SPI activities are *organised*, 2) all improvement efforts are carefully *planned* and 3) *feed-back* on effects on software engineering practices are ensured. The approach to SPI initiatives is guided by three additional ideas: 1) SPI is *evolutionary* in nature, 2) SPI is based on idealised, *normative* models of software engineering and 3) SPI is based on a careful creation and development of *commitments* between the actors involved. Finally, the perspective on the SPI goal is dominated by three ideas: 1) SPI is focused on software *processes*, 2) the practitioners' *competencies* are seen as the key resources and 3) SPI aims to change the *context* of the software operation to create sustainable support for involved actors.

4. The Case

This study was conducted at AstraZeneca, one of the world's leading pharmaceutical companies. AstraZeneca is a research-driven organisation with a formidable range of products designed to fight disease in important areas of medical need. The company was formed in April 1999 by the merger of Astra AB and Zeneca Group PLC. AstraZeneca has a strong research base and powerful product portfolio, designed in seven areas of real medical need – cancer, cardiovascular, central nervous system, gastrointestinal, infection, pain control and anesthesia, and respiratory. AstraZeneca is world number three (1999) in ethical pharmaceuticals and has more than 50,000 employees world-wide. There are research and development (R&D) centers of excellence in Sweden, UK and the USA and R&D headquarters in Södertalje, Sweden. The company has some 10,000 R&D personnel and a US $2 billion R&D investment in 1999, extensive global sales and marketing network, employing over 25,000 people, and 12,000 people employed in production in 20 countries.

4.1 The Software Organization

AstraZeneca has four departments that supply global IT services to the whole company: one in the UK, one in the USA, one in Sweden and one to provide IT support for research and development for the whole organisation. In addition to this, there are five global supplier managers who have the responsibility of controlling the needs of IT services in the business functions in the company. Furthermore, there is a company staff with central IT departments for solving problems related to technology adoptions, infrastructure, security, integration, and strategies. Beyond all this, there are IT functions that support the local marketing company in the respective countries. There are in total 2,500 persons working with IT-related questions in AstraZeneca.

This research started before the merger between the two companies in an IS organisation called Clinical Research and Information Management (CRIM) at the former Astra Hässle in Sweden and continued later in the new IS organisation, which then changed its name to Development IS (DevIS). DevIS supports clinical and pharmaceutical projects, Regulatory Affairs and Product Strategy and Licenses at AstraZeneca R&D Mölndal. DevIS is also responsible for influencing the development of the global clinical research processes and IS/IT tools in AstraZeneca. DevIS comprises 90 people including contractors, most of whom have backgrounds in IS/IT.

Many regulatory authorities require that pharmaceutical companies and their software organisations comply with GXP (Good *Manufacturing* Practice, Good *Clinical* Practice, and Good *Laboratory* Practice) rules. GXP rules are the authorities' quality requirements to pharmaceutical companies for ensuring patient health, the quality of processes (e.g. clinical studies or software development) and the quality of products (e.g. tablets or software). As a software organisation in the pharmaceutical business, DevIS must address many quality requirements. One fundamental requirement is that DevIS must be able to show the authorities, by documented evidence, that software development activities (e.g. software change control, software validation, and data processing and storage) are being performed in compliance with quality requirements. Therefore every software project regulated by GXP requirements should carefully apply all quality rules and be able to show by documented evidence that the software is compliant with the related GXP requirements. The company long ago adopted standard operation procedures that explicitly describe the company's software quality rules. These standard operation procedures should be applied for all information systems regulated by GXP requirements.

Employees of DevIS are basically engaged with software development, software maintenance and software operation activities. The software development activities occur in two forms: 1) development of totally new software products (software development) and 2) developing or changing existing software products (software maintenance). A typical software development project at DevIS is scheduled to take between six months and one year and includes analysis, design, construction, testing, and validation. Software maintenance activities can consist of changes in the code or developing a completely new application for existing software products. Software products in DevIS include the software and all related documentation (e.g. user requirement specification, test plan, validation plan, validation report, user manuals etc.).

4.1.1 The Problem Area

The results of a problem analysis performed in early 1999 in one of CRIM's largest software development groups showed a need for improving software project disciplines and providing guidelines to understand the standard operation procedures and GXP rules. The director of DevIS initiated an improvement project called Software Process Improvement at CRIM (SPIC) (whose name was changed to SPID after the merger (Software Process Improvement at DevIS)) to understand the existing problems and improve the organisation's software processes. The following figure illustrates a rich picture of the SPID project.

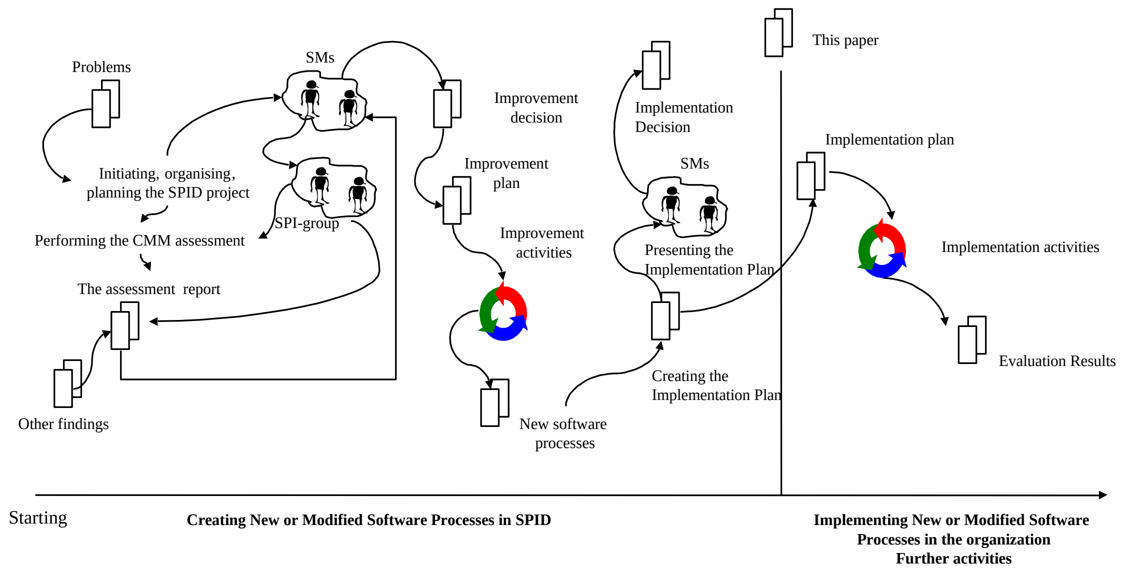

Problems
SMs
Improvement decision
This paper
Implementation Decision
Initiating, organising, planning the SPID project
Improvement plan
SMs
Implementation plan
Performing the CMM assessment
SPI-group
Improvement activities
Presenting the Implementation Plan
Implementation activities
The assessment report
Other findings
New software processes
Creating the Implementation Plan
Evaluation Results
Starting
Creating New or Modified Software Processes in SPID
Implementing New or Modified Software Processes in the organization Further activities

Figure 3 : The rich picture of SPID [18].

The SPID project was initiated, organised, planned, and performed during the period of April 1999 to May 2000 aiming to improve DevIS's software processes. A maturity assessment using a modified CMM-based (Capability Maturity Model) assessment method, QBA (see [19]), showed that DevIS was by then a level one organisation and addressed improvement possibilities in all analysed KPAs (Key Practice Areas). An improvement report based on the assessment's findings and other findings from earlier improvement initiatives at DevIS addressed six improvement activities. The steering committee of SPID gave priority to the following improvement activities (improvement decision) from the improvement report:

1. To establish a minimum documentation level for documenting the results of software projects and create the software documentation process.
2. To improve processes for software validation, software change management, and document version control.
3. To create a template library including templates for documentation of software development activities, such as: user requirement specification, design specification, test plan, and validation plan.

An improvement plan was created and the SPI group (including the author, two SPI consultants and five software engineers) started planning and performing improvement activities over a period of four months, which resulted in creation of new software process guidelines. An implementation plan was then created and presented for the steering committee. The steering committee of the project accepted the new software processes and decided to implement the newly created processes throughout all of DevIS. The implementation activities are scheduled to be arrayed out between August 2000 and June 2001. The implementation activities include among others a trainee program for all practitioners at DevIS and aim to change the context in which the new software processes will work. The implementation phase also includes further improvement activities in which the created processes will be improved on the basis of experiences of using them in practice. This phase will result in a new version of the software process guidelines in June 2001.

5. The Findings

In this section we discuss the knowledge creation and the learning process that took place during the creation of new software processes in SPID. Further we discuss the most characteristic features of SPI affecting the SPID project.

5.1 The learning process in SPID

According to Pourkomeylian [20] it is useful to view an SPI project as an organisational knowledge creation process in which certain type of created knowledge deals with the fundamentals of SPI (SPI-knowledge) such as: management of SPI initiatives, issues related to how an SPI initiative should be guided, and issues related to SPI's focus on target(s) (see [2]). Others deal with issues related to Software Engineering practice (SE-knowledge) such as: project planning, quality assurance, change control, and configuration management (see [21]).
In SPID the created SPI-knowledge was such that it was able to be put into practice during the project, while the created SE-knowledge could only be practised in a testing phase in which the new or modified software processes could be verified. This means that the learning process for the SPI-knowledge in SPID started when the project manager created tacit SPI-knowledge at the individual level in the socialisation phase to gain management and practitioner's commitment to the project. At that time the level of SPI-knowledge in the organisation was very low and the management was not able to see the organisational benefits of an SPI project. Thus the project manager arranged meetings with both management and the SPI-group to explain the concept of SPI and the benefits that an SPI project might give to the organisation. The created SPI-knowledge transformed afterwards very slowly via other practitioners in the organisation in one on one talks during coffee breaks as well as formal and informal presentations by the project manager in different meetings. The created tacit SPI-knowledge then becomes explicit SPI-knowledge through dialogue in the externalisation phase in which conceptualised SPI-knowledge was created primarily by the project manager and the steering committee. This knowledge was in the form of project specifications including plans, schedules, roles, and responsibilities. The created explicit SPI-knowledge was then combined with other knowledge or experience of improving software processes or other improvement models and/or methods in the combination phase in which systematic knowledge was then created. In this phase, knowledge was created mostly at the individual and group levels and all actors were involved in the knowledge creation process. The created explicit SPI-knowledge was then put into practice by all actors involved in the SPID to make SPI happen in the project. Operational knowledge about the SPI was created by practising SPI activities in the project. We learn about the SPI approach by doing it in practice. In this phase, the created explicit SPI-knowledge became tacit SPI-knowledge through practice. This means that the learning process in SPID took place almost through all OKC phases involving all actors in individual and group levels and led to creation of SPI-knowledge.

5.2 The management of SPID

SPID was organised as a project with specific budget and resources. The initial improvement infrastructure of the project was established early in the project, and the roles and responsibilities in the infrastructure were described. The organisation of SPID consisted of: one project manager (the author of this paper): responsible for co-ordinating, planning and performing the project; a steering committee: including the software managers and the director of DevIS responsible for allocating resources to the project and deciding on the acceptance of the results of the project; a reference group: including software engineers and project managers responsible for giving input to the software improvement activities for creating the new software processes; and a working group: including the SPI consultants responsible for documenting the newly created software processes. An SPI plan was created to define the deliverables, milestones, schedules, and goals of the project. Feedback on improvements was not defined explicitly. It was simply mentioned in the SPI plan that the results of the project (the new software processes) should be tested in two software projects before implementation in the whole organisation.
Organising the SPID as a project gave us the possibility to allocate resources to the project as with any other project at DevIS. This helped to gain the management's commitment to the project during the project's entire life span. It further helped us to structure the project's organisation and the responsibilities. Organising SPI initiatives as regular software projects has been supported in earlier work by [7], [2], and [22]. Organising SPID as a project brought risks as well. People working in the reference group were very busy with other projects at the same time and this sometimes meant that

someone could not deliver what he/she was supposed to deliver at a meeting. However this was a minor problem. Co-ordinating the resources and the meetings, both with the management and the improvement team, demanded a grate deal of planning time. On the other hand, planning the project helped us to understand: what to do, when to perform which activity, and who should do what in the project. This helped us to focus on our schedule and the deliverables. This sometimes caused stress, especially for the project manager. Another risk of detailed planning was that we sometimes felt that we were prisoners in the schedules and deliverables, although we did not let our thoughts and ideas be stopped or disturbed by the depth of the plans. We tried to “reflect-in-our-actions” all the time and changed some plans and some deliverables a couple of times. These changes were either necessitated by other ongoing improvement activities in the company which had effects on our project or because we felt that some changes in a specific plan and its deliverables would provide better results for the project in the end. Not defining the feedback on the improvements in detail actually did not affect the initiation of the SPID, but we missed some feedback during the improvement activities in terms of knowing whether we were on the right path.

5.3 Approaches to the SPID

In SPID we decided to improve a few software processes in an evolutionary way by taking one step at a time. The goals were neither to reach any maturity level in the CMM nor to improve several software processes. To understand the current level of software process problems we adopt and performed a modified CMM-based assessment, which helped us to identify our software process problems in a structured way. Because the goal of SPID was not to reach any maturity level in the CMM we decided not to follow the CMM recommendations in our improvement activities. We rather let ourselves get inspiration from the concept of SPI, i.e. improving software processes in a systematic way, than tried to fulfil the CMM’s requirements. We based our improvement strategy on the assessment’s findings, other quality goals within the organisation, and practitioners’ and management’s commitment to the project. Without the management’s commitment we could neither start the SPI initiatives nor get the necessary resources for the project and without the practitioners’ commitment to the project we could never have constructed a creative forum for discussion and improvement of the new software processes.
Having an evolutionary approach in SPID helped us to concentrate on a few software processes. We could see the whole picture and did not get lost in the complexity of having several software processes to improve. We could manage the situations well and had time to reflect in our actions to improve our SPI practice. The problem with focusing on a few software processes was rather that the software processes focused on were related to other software processes and software development models. This caused many hours of discussion to separate other issues from our main scope. We still however needed to remind each other of the scope of the software processes several times during our project. Not aiming to reach a maturity level in the CMM led to a positive reaction among both the management and the practitioners. Because the goal was not just to reach an “abstract” target (reaching a “level” in a model, for management and practitioners at that time) but, solving the organisation’s problems by being inspired by a well-known model, which was modified to suit the organisation’s situation. This created more motivation and enthusiasm among practitioners and management and strengthened their commitment to the project. Being dependent on both management and practitioners’ commitment was one of the most vulnerable key issues of the SPID. This meant that, before starting the project, the project manager had to carry out many marketing activities for the project to gain both the management’s and the practitioners’ commitment and this was a time demanding activity. One key factor in succeeding in creating new processes was that the practitioners believed in SPID and were considered about the results of the project. Practitioners working with SPID were almost all persons with in-depth experience in leading software projects and wrestling with software process problems. They knew the importance of solving the software process problems, and this made them very committed to the project and involved in working hard to create processes that could function in practice.

5.4 Perspectives in SPID

At that time DevIS did not have a detailed description of all software processes. This meant different interpretations of any given simple software process activity in the software projects. For instance, the practitioners knew that a software product should be validated before being put into operation, but the interpretation of the software validation process was different in different projects. We knew from earlier improvement activities that a description of the software documentation process was missing in the organisation. Knowing which documents should be created as products of the software projects could help us to identify the activities needed to perform for creating them. It could further help us to identify the main processes needed for supporting these activities. We therefore adopted a product perspective in the beginning of the project and focused on identifying the documents needed as the results of a software project. On the basis of this, we identified actors and activities needed for creating each document. After defining the software documentation process we changed our focus and concentrated on processes that should be improved (software validation and software change control). During the improvement process the practitioners' competencies, ideas and experiences were the main input to the improvement work. Because the improvement phase of the project ended before the implementation phase we did not need to change the context in which the software processes should be operated.

The problem in focusing on software processes was that these processes were related to the other processes, the software development models, and the software project management models. It required many hours of discussions to separate different issues from the project's main target and focus on the defined goals. The benefit of focusing on documents as software product was that we agreed upon the documentation as one type of software products. This helped us later to identify the activities and processes needed to be in place to support the creation of these documents. All the improvement activities in SPID were based on the practitioners' ideas and experiences. One problem of focusing on practitioners' competencies for improving software processes was that the whole project was dependent on their input. If a majority of practitioners was not able to join a meeting we cancelled the meeting and had to wait until the next time. This problem has been identified by Johansen and Mathiassen [7]. Another problem was that the practitioners had different experience from different software projects. This caused variations in interpretation of any specific software process activity. Much time was needed to discuss different views and experiences related to one specific software activity or process. However the benefit was that we knew that all the different issues discussed had already been put into practice and shown some degree of usability. Table 2 illustrates the SPID activities based on the MAP described by Aaen *et.al* [2], *: Total, ~: Partly, -: Nothing.

Concern	Idea	Performed	Description	Aspiration	Pitfalls
Management of SPI	Organisation	*	The SPID was organised as a project like all other software projects within DevIS having specific budget, and resources. The organisation of SPID consisted of a reference group, a steering committee, a working group, and a project manager.	Allocated resources. Gaining management's support. Structuring a formal organisation for the project.	Inadequate resources. Difficulties co-ordination.
	Plan	*	An SPI plan was created to identify the millstones, deliverables, responsibilities, and activities needed to be performed within the project.	Knowing what to do, when to perform which activity, and who does what.	Stress feelings. Bounded in plans and deliverables.
	Feedback	-	The SPID didn't defined in detail how feed back on improvements should be measured.	Saving time.	Could not measure quality during the project.
Approach to SPI	Evolution	*	SPID aimed to learn from the SPI concept and improve a few software processes in an evolutionary way by: Identifying the current level of the software process problems Identifying the minimum documentation level Creating processes for: Change management, and Software validation	Learning by doing. Stepwise improvement. Adapting the concept of SPI by reflecting-in-action. Not getting lost. Could see the whole picture.	Not having the whole picture. Time demanding.
	Norm	~	Using a modified CMM-based model only for the assessment to identify the current maturity level of the software organisation. Not aiming to reach any maturity level in CMM. Basing the improvement strategy on the assessment findings and earlier improvement findings.	Identifying the software process problems in a structured way. Using a modified version of a well-known model. Gaining management and practitioners' commitment to the project.	Took a long time to understand that we still can learn and use the concept of SPI without blindly following the CMM.
	Commitment	*	One important factor in getting succeed with SPID was both practitioners' and management's commitment to the project. During the whole project we had the management's support and practitioners' commitment to collaborate in creation of new software processes.	Ensured the resources for the project. Ensured dedication	To be dependent on management and practitioners in two different ways. Time demanding marketing activities.
Perspective in SPI	Process	*	In the beginning SPID was focused on the products (documents) of the software projects. This focus changed later on in the project to the processes.	Two targets with one shot, i.e. defining processes by identifying products.	Having the scope in focus all the time.
	Competence	*	The main input to SPID was practitioners' ideas, and experiences about the software processes.	The practitioners' ideas and experiences were tested in the reality.	Being dependent. Different views.
	Context	-	During the SPID the context in which software processes operated was not changed.		

Table 2: SPID activities based on the MAP see [2].

6. Lessons Learned

We have analysed an SPI project based on the most characteristic features of SPI (see [2]) and an OKC perspective using a framework based on Nonaka and Takeuchi [15] of the organisational knowledge creation process. From the perspective of this framework we believe that: a novice organisation has a chance to succeed with its first SPI initiation if the organisation maximises its ability to learn from the concept of SPI, and minimises the amount of software processes to be improved. This study suggests a number of lessons relevant for future SPI projects and the SPI practice.

Lesson one: *The first SPI initiative should aim to learn the concept of SPI and the software process problems.* An SPI initiative is a complex effort in which people, methods, tools, and products interact with each other. The first SPI initiative in a novice organisation should therefore be a learning process in which the organisation learns about the concept of SPI and the current software process problems while improving software processes.

Lesson two: *A novice organisation should focus more on the SPI concept than the CMM recommendations.* The first SPI initiative should start by diagnosing the current maturity level of the organisation. The CMM can offer much help in doing this. However for improving the software processes, it is better to rely on the organisation's goals and focus on the SPI activities and learn to improve a few software processes based on the organisational goals rather than just following the CMM as a model.

Lesson three: *The first SPI initiative should be organised as a project, with specific goals, deliverables, and recourses aiming to improve a few software processes in an evolutionary way, on the basis of the organisation's ideals and practitioners' ideas.* Organising the SPI initiative as a project and planning the activities are essential for managing improvement activities in an SPI initiative. For guiding the SPI improvements it is much more important to focus on stepwise improvement and satisfy the organisation's goals than to follow an abstract goal. This supports gaining the management and practitioners' commitment to the project. An SPI initiation should be focused on the practitioners' ideas and experiences for improving the software processes.

These lessons correspond with [2], [7], and [22]. However, SPID did not adopt the whole MAP in detail. But as the first SPI initiative it is most essential to be focused on delivering results that are visible to the management in the scheduled time. This will ensure dedication and legitimacy for the continuous SPI efforts in the organisation. As a success criterion it should be mentioned that the steering committee of SPID accepted the new created software processes and decided to implement these processes in the whole organisation. The implementation phase in SPID starts in August 2000.

As mentioned before one great challenge for DevIS is to find a way to implement the newly software processes in the organisation. A key factor in helping the implementation of the created new software processes is training. According to the implementation suggestion report presented to the steering committee of SPID all practitioners working with software development and maintenance should have training in the new processes. This means that, based on the trainee program, each practitioner will create his/her own understanding of the newly created software processes. These understandings may not be the same because of the differences in practitioners' experiences and backgrounds. Because of this, as well as other factors that might cause variation-in-action, DevIS should expect variations-in-action (see [23]) when these new processes are put into practice. Questions for further research might then be: 1) Does variation-in-action happen in DevIS using the new software processes? 2) How does it happen? 3) Which factors are involved in affecting variations-in-action? 4) What does it mean for DevIS? 5) How does variation-in-action affect the institutionalisation of the new software processes at DevIS?

7. Acknowledgement

This research was sponsored by AstraZeneca R&D Mölndal in Sweden in collaboration with the Viktoria Institute and the Institute of Computer Science in Aalborg, Denmark.

8. References

[1] Humphrey, Watts S.: Managing the Software Processes, Addison-Wesley Publishing Company, USA. 1989.

[2] Aaen I., Arent J., Mathiassen L., Ngwenyama O.: A Conceptual MAP of Software Process Improvement. Artikelsamling, Center for Softwareprocesforbedring. Dansk Elektronik, Lys & Akustik. 2000.

[3] Tryde, S., A.-D. Nielsen & J. Pries-Heje 82000). A Framework for Organizational Implementation of SPI in Practice. In: L. Mathiassen *et al.* (Editors): *Learning to Improve.*

[4] Goldenson, D. R., and J. D. Herbsleb.: After the Appraisal: A Systematic Survay of Process Improvement, its benefits, and Factors that Influence Success. CMU/SEI-95-TR-009, SEI, Pittsburgh. 1995.

[5] Herbsleb, J., *et al.*: Software Quality and the Capability Maturity Model. Communications of the ACM, 40(6), 30-40. 1997.

[6] Mashiko, Y., and V.R. Basiili: Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement. Journal of Systems and Software, 36. 17-32. 1997.

[7] Johansen, J., and L.: Mathiassen. Lessons learned in a National SPI Effort. EuroSPI 98 Göteborg. 1998.

[8] Aaen I., Arent J., Mathiassen L., and Ngwenyama O., Mapping SPI Ideas and Practices. 2000, TBD

[9] Paulk Mark C., *et al.*: The Capability Maturity Model for software Version 1.1, Carnegie Mellon University, Software Engineering Institute. 1993.

[10] Kuvaja Pasi, Bicego: BOOTSTRAP - a European assessment methodology, Software Quality Journal 3, 117-127. 1994.

[11] McFeeley Bob: IDEAL: A User's Guide for Software Process Improvement, Software Engineering Institue, Cornegie Mellon University. 1996.

[12] Zahran Sami, Software Process Improvement, Software Engineering Institute, SEI Series in Software Engineering, Addison Wesley Longman 1998

[13] Curtis, B., A Mature View of the CMM. American Programmer, 7, 9, 19-27. 1994.

[14] Patton M., Q., Qualitative Evaluation and Research Methods. SAGE Publications. 1990.

[15] Nonak, I., and Takeuchi H.: The Knowledge-Creating Company, Oxford University Press, Inc. 1995.

[16] Polanyi, M., : The Tacit Dimension. London: Routledge & Kegan Paul. 1966.

[17] Nonaka I. And Konno N.: The Concept of "Ba": Building a Foundation for Knowledge Creation, California Management Review Vol 40, No.3, Spring 1998

[18] Checkland Peter, Scholes Jim: Soft System Methodology in Action, John Wiley & Sons LTD. England. 1990.

[19] Arent J., Iversen J., Development of a Method for Maturity Assessments in Software Organizations based on the Capability Maturity Model, in Department of Computer Science. Aalborg: Aalborg University, 1996.

[20] Pourkomeylian P., Knowledge Creation in Improving a Software Organization. IRIS 2000.

[21] Pressman Roger S.: Software Engineering, a practitioner's approach, fourth edition, International editions. 1997.

[22] Aren J. and Norbjerg J: SPI as Organizational Knowledge Creation: A Multiple Case Analysis. Proceedings Conference Proceedings at Maui Hawaii, January 2000.

[23] Schön Donald A., Educating the Reflective Practitioner. Jossey-Bass Publishers . San Francisco. 1987.

Instilling Quality Improvement

Joshua Klein
Yigal Cohen
NDS Technologies, Israel

Abstract

NDS provides systems and services for managing and controlling the secure distribution of entertainment and information over digital broadcast media and the Internet. Competition forces NDS to diversify its services, anticipate market needs, deliver what was promised, and do so all at a reduced cost.
In response to these pressures, NDS management requested the QA staff to improve requirements management, the first link in the development chain. Given production and marketing demands, the resources invested in quality infrastructure were limited. In addition, there were very few QA engineers available to work on improving processes. The QA team decided to take a less academic, more pragmatic, approach to advancing quality processes with minimal top management support.
The QA team shifted its approach away from ISO 9000's formal definitional style to a people oriented style of change management. This approach was enthusiastically received by developers and mid-level managers who showed significant interest and willingness to invest in more structured methods of work.
All that being said, pressures to produce and market under tight schedules limited the establishment of quality improvement practices. Nevertheless, NDS did experience progress in its requirements management due mainly to the real responsiveness of developers and mid-management.

Description of NDS

NDS is a leading supplier of open conditional access software systems and interactive applications to digital pay TV broadcasters and set-top box manufacturers. NDS's open and flexible software solutions provide the secure delivery of entertainment and information to TVs and PCs, over cable, satellite, terrestrial and Telco links.

Change from the Bottom

This Quality Assurance story begins from the bottom, rather than from the top.

NDS developers felt frustrated by the lack of clarity and precision in the requirements documents they received. If the requirements were fuzzy, then the code would be fuzzy as well. And fuzzy code leads to unsatisfied customers.

Management heard the developers' complaints, and asked the Quality Assurance (QA) department to teach the technical writing staff how to write better requirements.

The QA staff sensed that the core of the problem lay elsewhere, and brought in consultants to fully analyze the problem. They consultants proposed a new way of thinking about the requirements process, which lead to many different types of processes than were currently in place.

But unlike other companies, where management reads about a hot new theory, and imposes their plan on the unsuspecting workers, at NDS the cry for change came from the developers themselves.

Attitudes within the NDS Culture

> “Culture can be defined as a pattern of shared basic assumptions that the people in an organisation have learned as they have solved the problems of relating to the outside world and of learning to work together. These assumptions lead to solutions that work well enough to be considered valid. Culture change can be defined as a change in the pattern of basic shared assumptions.” [1]

Management

The NDS management did not believe in fashionable improvement theories and methods. They wanted to see visible, short-term results. In order to capitalize on the developers’ momentum, management established two committees: a steering committee and a process group.

Developers

The developers and mid-level managers were enthusiastically supportive of the new QA efforts. They *wanted* to move to a more structured method of work.

Concurrently, they felt stress from their tight deadlines, and pressure from their perception of management expectations. [2]

People Rather than Process

The focus of the QA department shifted from a formal definition of processes, which is the most important issue in ISO, to the people who needed to adopt a new and improved way of running the development cycle [3].

Plans and Expected Outcomes

The QA Team

Two outside consultants were brought in to support the NDS QA efforts: an organizational expert, who is psychologist, and a software engineering expert.

The Goal

NDS sought a new way to improve the requirements process in order to:

- Produce reliable software code

- Improve customer service
- Increase productivity
- Anticipate market needs
- Reduce expenses by “getting it right the first time”
- Remain competitive in a market that is catching up to its lead.

The Plan

The consultants first performed an assessment of NDS. Based on their findings, they devised the following three-phased plan:

1. Conduct pilots

   During the pilots, high-level NDS employees, with the support of the consultants, tried using different methods than the ones currently in place at NDS, focusing on non-production related items. The employees defined a challenge, found appropriate tools, customized them for the NDS setting, and then evaluated the applicability of the new tool for NDS.

2. Build infrastructure

   The infrastructure was required to support the new culture. This included:

   - Requirement Management tools
   - New methods and the related training
   - Requirements books and periodicals made available in the library
   - Users groups
   - Intranet site
   - Reward system

   The infrastructure was developed at the same time that the pilots ran, to support the needs of the pilots. When the pilot phase ended this infrastructure became part of the global NDS infrastructure (during the Institutionalization phase.)

3. Institutionalize quality efforts throughout the company

   - Expand the pilots to real projects
   - Make the Requirements Workshop a pre-requisite for all those involved in crafting requirements documents.
   - Prepare a one-time presentation about Requirements Management concepts, to be presented at each department’s monthly meeting.
   - Coach all projects through each requirements phase.
   - Track and measure the spread of the new requirements culture throughout the company.

   The Requirements Plan can be seen in Figure Fig. [JKLEIN] 1 below:

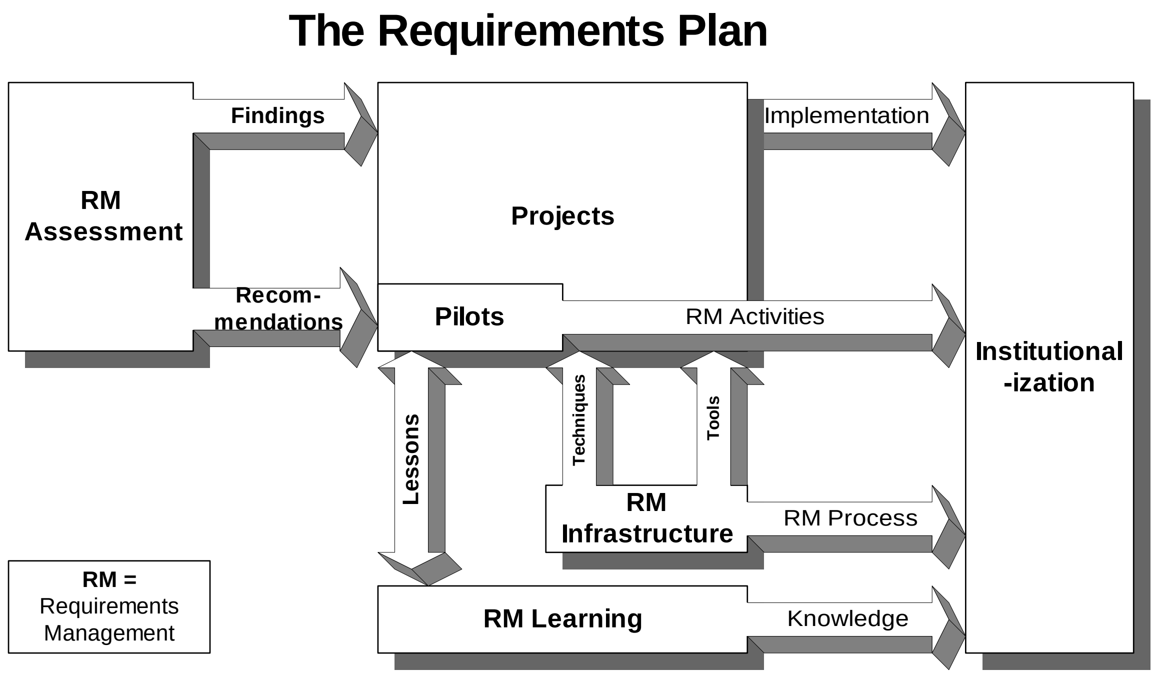


Fig. [JKLEIN] 2: The Requirements Improvement Plan

Implementation

Pilot Activities

Three of the following five pilots were completed during the testing phase:

1. No actual customer (anticipating future market needs)
2. Use of a requirements management tool
3. Top-down requirement decomposition
4. RFP (Request For Proposal)
5. Peer Review

Pilot #1: No actual customer

Description	Notes
Requirements Problem	Define system functionality based on market and business applications.
Pilot Goal	Verify functional requirements with respect to defined business applications - possibly subset of all applications

Description	Notes
Pilot Activities	Peer Review Update document per Action Items generated by Peer Review

A sales person from the marketing department ran this pilot.

It focused on the issue of "requirements coverage:" whether the current set of requirements that the marketing department defined (there is no actual customer) was comprehensive. This was determined by checking the marketing requirements in contrast to the business functional requirements. It was important to the sales people to convey the marketing needs to the research and development engineers.

During the pilot, the sales person ran a simulated peer review in the process group as well as a real peer review with the true stakeholders.

The reviews uncovered important holes in the existing set of marketing requirements. We learned that even when there is no external customer, NDS employees can provide a great deal of information about customer needs.

Pilot #2: Use of a requirement management tool

Description	Notes
Requirements Problem	Too many requirements to track manually
Pilot Goal	Requirement change impact and assistance for the existing system
Pilot Activities	Capture data using Rational Requisite Pro. Input the relevant requirements specifications ♦ Trace links between components and requirements listed in the Rational DDTS examples ♦ Add full traces between requirements and components for specifications entered ♦ Generate report of change process impact

A senior technical writer ran this pilot.

Requirement engineering was allocated to the NDS technical writers because they are the ones who compose the requirements documents, in conjunction with subject matter experts. Making the technical writers responsible for tracking the requirements (instead of the engineers) allowed the engineers to return to their engineering work.

Requisite Pro provided the technical writers (and students, who do the actual data entry) with a way of transforming a simple MS Word document into a formal database. It also provided a means of tracing between requirements.

Pilot #3: Top-Down Requirement Analysis

Description	Notes
Requirements Problem	Derive product requirements from the end-to-end requirements
Pilot Goal	Define testable and traceable end-to-end and product requirements
Pilot Activities	♦ Program Manager and Marketing: Request for Proposal (RFP) and requirements draft when relevant ♦ Project team: Create end-to-end and product requirement documents when business status is clearer ♦ Define the links between both of them.

A project manager ran this pilot. This particular project manager was chosen because he previously was an NDS system engineer.

NDS not only has a top-down organizational structure, but also a horizontal flow that starts in the Projects axis of the organizational matrix and ends in the Development axis.

This pilot had difficulties do to the lack of requirement engineering education at NDS.

Pilot #4: Request for Proposal (RFP)

Description	Notes
Requirements Problem	Receive an RFP from a customer and return with Business Requirements
Pilot Goal	Involve the internal stakeholders in building the business requirements
Pilot Activities	♦ Get a formal RFP from the customer ♦ Present the most recent NDS technology ♦ Take NDS development constraints into account

A project manager ran this pilot. It involved a real RFP from an outside customer.

Initially, the QA department welcomed the allocation of resources that this project was granted. Unfortunately, because the customer cancelled the project altogether, the QA pilot was also terminated.

However, the ideas developed during this pilot contributed to the final requirement process map.

Pilot #5: Peer Review

Description	**Notes**
Requirements Problem	Correct data at an early stage
Pilot Goal	Have peers involved in document reviews in an effective, resource-saving way.
Pilot Activities	♦ Run mini reviews with the process group. ♦ Define or acquire forms, samples and training.

A Senior Technical Writer ran this pilot.

Peer Review is a technique that is not exclusive to Requirements, but is valuable for improvement in general. It is practice process SUP.5 and generic practice 3.2.2 in ISO 15504 Part 5.

In general, peer review improves the effectiveness of meetings, more than a trial of a method specific to requirements.

Numerous NDS employees use the forms that were developed for this pilot, but we are still far for thoroughly taking advantage of this important technique.

Infrastructure Activities

1. Requirements Management Process

   NDS created a requirements process as described in Figure Fig. JKLEIN 3 below:

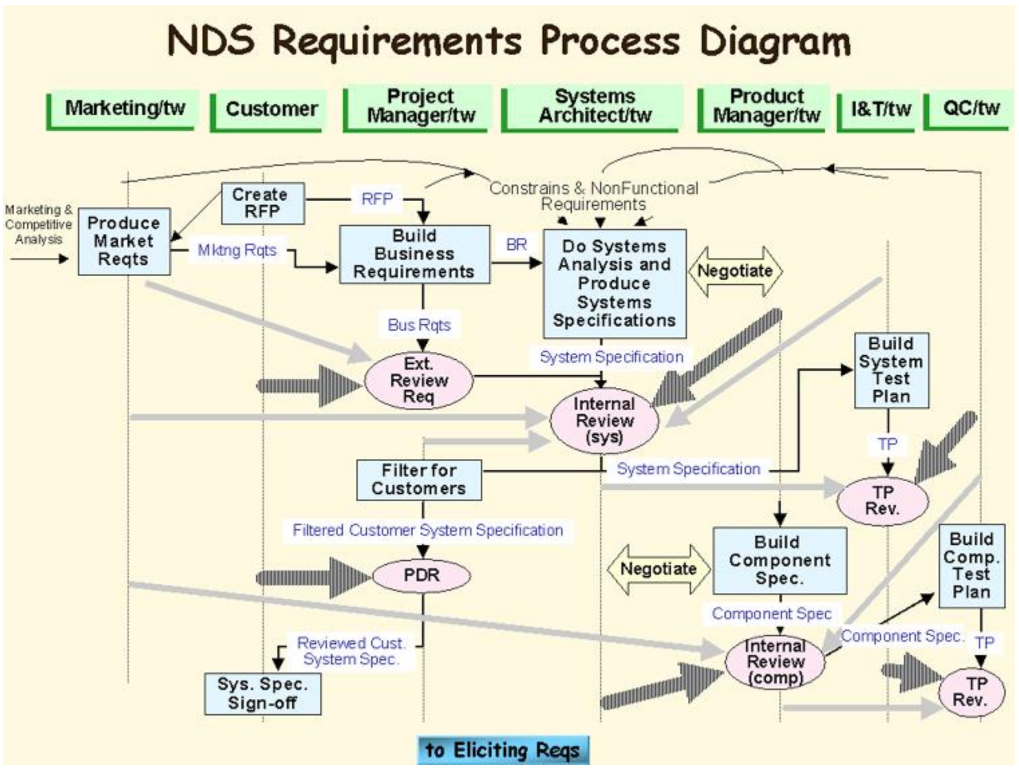


Fig. JKLEIN 4: The Requirement Process Diagram

2. Requirements Workshop.

   The Requirements Management process in Figure Fig. JKLEIN 5 above has been taught to over 100 participants who attended a Requirements Workshop. During each workshop, a vice president comes to the course to meet the participants. He solicits suggestions from them, which he then brings to the steering committee. This is another example of the bottom of the pyramid driving the top.

3. Rational's RequisitePro Tool

   The tool chosen for requirements management was Rational's RequisitePro. This tool is used by eight projects (40 employees) to track requirements, create traceability matrices, and to generate reports that span the Business Requirements down to the Test Cases performed in the lab.

4. The RequisitePro Users Group

   Another “bottom-initiated” activity at NDS includes the RequisitePro Users Group. This group meets regularly in order to share the challenges and lessons of Requisite Pro. It is chaired by a manager whose project was one of the first to participate to the deployment phase.

Institutionalisation Activities

The NDS QA process builds upon the grass-roots support that exists for cultural change. Quality practices are instituted from the “top,” and then spread by the “bottom,” horizontally.

The following hurdles were overcome: [4].

1. Ignorance: This was remedied by the Requirements Management Course, by information spread over the Intranet, and by the availability of books and magazine in the library.
2. Lack of Time / Resources: Even when employees were not ignorant of the knowledge or best practice, they lacked the money, time, and management resources to pursue and study it in enough detail to make it useful This was remedied by the steering committee’s tracking, measurement, official management support.
3. Lack of Pre-existing Relationships between Managers: People absorb knowledge and practices from other people they know, respect and - often – like. If two managers have no personal bond, they are less likely to incorporate each other’s experience into their own work.

   This was remedied by: the department kitchenettes, which serve the purpose of allowing employees to informally exchange information and experiences, and the Quality Assurance moderated bulletin board
4. Lack of Motivation: This was remedied by instituting a reward system, and by holding debriefings after each project was over.

Results and Lessons

What are the measured results of all of these audits and QA efforts?

Of the 20 NDS projects surveyed:

- 40% of the documentation required by the new process was written
- 60% of the stakeholders play the role they are expected to
- 45% of the reviews are held according to the process expectations
- 65% of the target population has attended the Requirement Workshop
- 20% of the requirements were ported to Requisite Pro

What have we learned so far?

1. The NDS management learned about how to implement cultural change at a company-wide level.

   Before the process began, management believed that a few books or course can solve the “quality” problem, because it’s just a matter of supplying knowledge or skills. It was a revelation to realize that this requires a cultural change from within the whole company in order to move from state (a) to state (b).

2. NDS learned that the quality of requirement is not just a technical writer issue, but is connected to the entire requirements management process.
3. Employees work more effectively when their functions and related responsibilities in the project chain are better defined.
4. Early meetings with customers should be attended not only by marketing personnel, but also by research and development engineers who can assess feasibility, and who can then immediately begin working on development.
5. Tests must be based on customer requirements, rather than on system design. In order words, the tester checks whether the product behaves the way the customer expects it to, rather than the way the system architect envisioned it.

Future Plans

- Intensify coaching of NDS employees who work with requirements. The NDS course is taught in 1 ½ days, rather than over the course of weeks. Therefore, extra coaching is needed to supplement the course.
- Focus on pre-sales (the high end) and on components (the low end). The primary focus had been on the middle of the chain: Business Requirements and the System Specification.
- Increase the number and variety of measurements. This will indicate how much change has occurred, and will point to where to invest further efforts. In the long run, measurements testify to the relationship between quality improvement and business results, which is what management wants to see.
- Teach requirements engineering as well as system engineering in-house.
- Acquire a richer Requirements Management tool that supports the process more comprehensively than Requisite Pro does.

Authors' CVs

Joshua Klein is quality assurance staff engineer at NDS Technologies Israel. He runs several improvement processes. He has contributed to the building of a set of development procedures in the spirit of ISO 9000-3. He also introduced several kinds of QA and QC tools. He has previously defined an original O.O. methodology for the Sapiens application generator and published some papers about it, mainly in French. He learned Mathematics, Physics and Computer Science at the Hebrew University of Jerusalem. He is a certified ISO Lead Auditor and successfully completed a project management course of the Israeli Institution for Productivity. Contact him at j.h.klein@ieee.org.

Yigal Cohen manages the quality assurance department at NDS Technologies Israel. He has an M.Sc in Computer Science with honors from the Hebrew University of Jerusalem.

NDS Technologies Israel

NDS is a leading supplier of open conditional access software systems and interactive applications to digital pay TV broadcasters and set-top box manufacturers. Our open and flexible software solutions provide the secure delivery of entertainment and information to TVs and PCs, over cable, satellite, terrestrial and Telco links.

NDS is also a leading developer and provider of open solutions that easily integrate for managing and operating digital television networks. NDS has a wide range of products that can be implemented from headend to set-top box and are complemented by a wide variety of services, including consulting, systems design and integration, support and maintenance.

NDS is headquartered in the U.K. and has offices in the U.S., Spain, Australia, Japan, Hong Kong and Israel. NDS employs over 1000 dedicated professionals - 300 of whom are dedicated to research and development.
In the past few years, NDS has won many awards for technology including an Emmy from The National Academy of Television Arts and Sciences and three Queen's Awards, one for Technology and two for Export.

References

[1] Kim Caputo, CMM Implementation Guide, ISBN 0-201-37938-4, p 20, Addison Wesley Longman, 1998

[2] Robert L. Glass, How not to prepare for a consulting assignment, and other ugly consultancy truths, Communications of the ACM, December 1998, Vol. 41, No, 12, pp. 11-13.

[3] Jim Van Buren and David A. Cook, Experiences in the adoption of requirements engineering technologies, CrossTalk The journal of defence software engineering, December 1998, pp. 3-9.

[4] Gabriel Szulanski, Intra-firm transfer of best practices project, American productivity and quality center, October 1994.

Challenges for software developing companies in the 21st century

Bernhard Kölmel
FZI Forschungszentrum Informatik Karlsruhe, Germany

Introduction

This article presents the results of an empirical study of the German ESPINODE (ESPINODE = European Software Process Improvement Node partially financed by the European Commission within the ESSI initiative) “IMPROVE software” concerning the main technological and process-oriented challenges for software developing companies. In this survey 156 companies answered questions on goals of their software process improvement (SPI) activities, the most important technological challenges, the processes to be improved, the main barriers, and needed assistance. It showed that the lack of qualified personal influences most of these topics, leading to time pressure and the lack of information exchange. While the technological goals of the companies are up to date, the process improvement goals seem to be the same for years. The potential benefits of software process improvement projects need to be much more communicated.

Importance of Software for Economy

Servicse and products, as well as the future of the industrial world are based more and more on software. The contribution of software for the added-value-creation-process is increasing everywhere and amounts, for example over 80% in the telecommunication industry [3]. The rising influence of software in all areas of industry, economy and in ordinary life is the reason for the ever-increasing demand of complex software. „*North American software spending enjoying six years of 20%+ growth*“ was reported by [6]. In the US [7] illustrates that “*beginning in 2000 the industry’s contribution to the [...] will exceed that of all other manufacturing industry groups*”. Fig. BK 1 demonstrates the expected growth software industry against the automobile industry in the US [7].

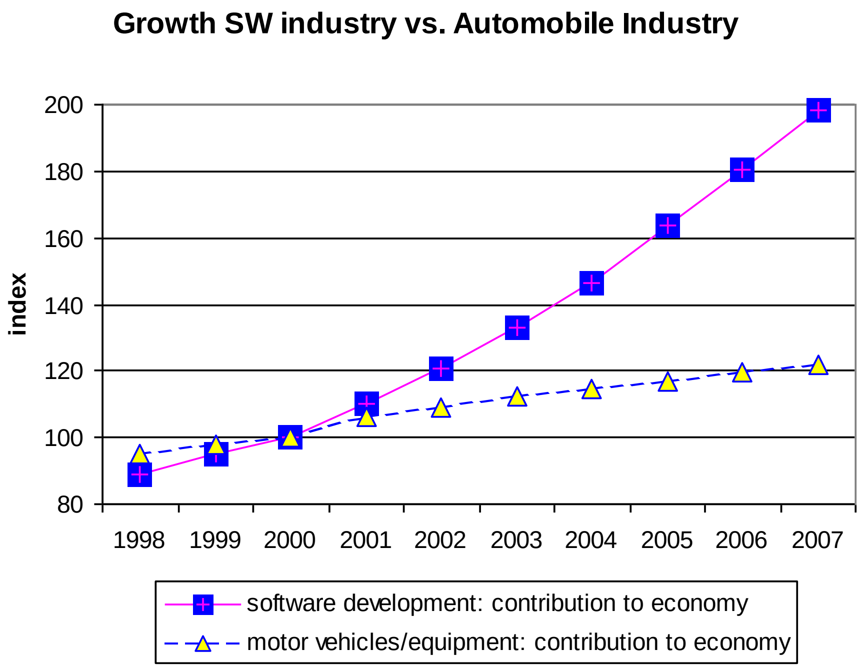


Fig. BK 1: Relative growth prospects of software industry vs. automobile industry in the US

The software developing organisations are facing a lot of problems. In accordance to a study of the Standish Group it becomes clear, that the level of software development have has to be improved. [5]: hardly 16% of all software projects are successful (i.e. time- and budget conform with all specified functionality). Some 53% are completed with problems (i.e. with enormous overtime which is in average 222% and much a 180% higher budget than calculated and/or substantial functional or quality impairment) and over 31% of the projects are cancelled at all.
Based on this the European Software Institute concludes that "*software process improvement is not just an admirable goal in itself but also a commercial necessity*" [8]. Nevertheless a lot of companies do not invest in software process improvement activities [4]. One reason might be that the effects and the return of investment of software process improvement activities are not widely known. Rementeria complains that "*too many of the benefits of process improvement appear not be obvious*" [9].
Another reason is that the provided services and assistance by consultants and tool-providers is not completely compliant with the requirements of industry. In an SPI-event one attendee complained "*the SPI consultants do address our needs, they want to sell their pre-defined approaches*".Too many software process improvement (SPI) consultants try to sell their defined approaches to software developing companies, but a concentration on topics which are needed by the companies is essential for success. "*An assessment usually identifies more improvement opportunities than you can tackle at once. "Focus" is a key word in process improvement.*" [Wie99]. It is necessary to identify the most challenging technological and process-oriented problem areas and concentrate the provided assistance in a few well-defined and well-needed improvement activities.
Therefore FZI Forschungszentrum Informatik Karlsruhe, on of the German ESPINODEs "completed an empirical study concerning the main challenges of software developing companies. More than 150 companies participated in the study. Within the study, the technological and process-oriented challenges of software developing companies (almost equally distributed over all company-sizes) have been identified. The study contains quantitative figures and detailed information on the distribution of the responses from the software developing companies

Empirical Basis

The questionnaire was handed out in the first quarter of the year 2000 at ESPINODE events aiming for

the improvement of software development, which discussed both process-oriented (e.g. topics like CMM, testing, configuration management) and technology-oriented topics (OO, Java, XML ,etc.). Additionally the questionnaire was mailed to some 250 software-developing companies in Southern Germany. Besides a short information about the company, the questionnaire contained questions on the specific goals of software process improvement in the company, the processes and technologies that need to be improved, as well as the existing barriers and the demand for support. Overall there was a response of 156 questionnaires, which represents an adequate basis.
The answers are almost evenly distributed over the different sizes of companies, whereby the companies with 50 to 249 employees are strongly represented (see Fig. BK 2).

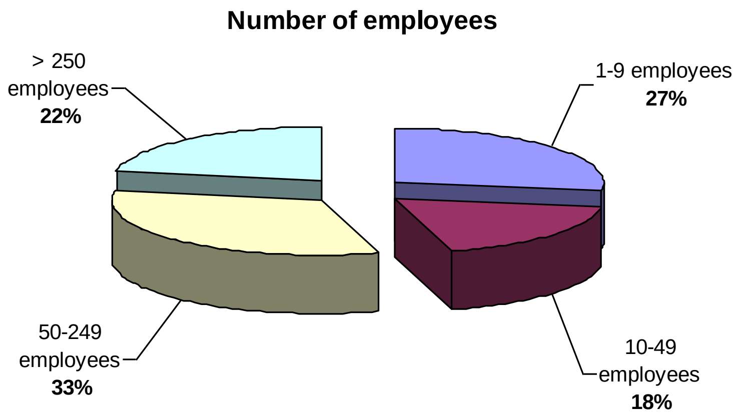


Fig. BK 2: Size of responding companies

The companies which completed a questionnaire represent all relevant market areas of software development (multiple-entry was possible), the most represented areas are individual development and dedicated software services. The topic of embedded systems is not very highly represented, although in Germany it is assumed that approximately 50% of all software development activities are in the scope of embedded systems (Fig. BK 3).

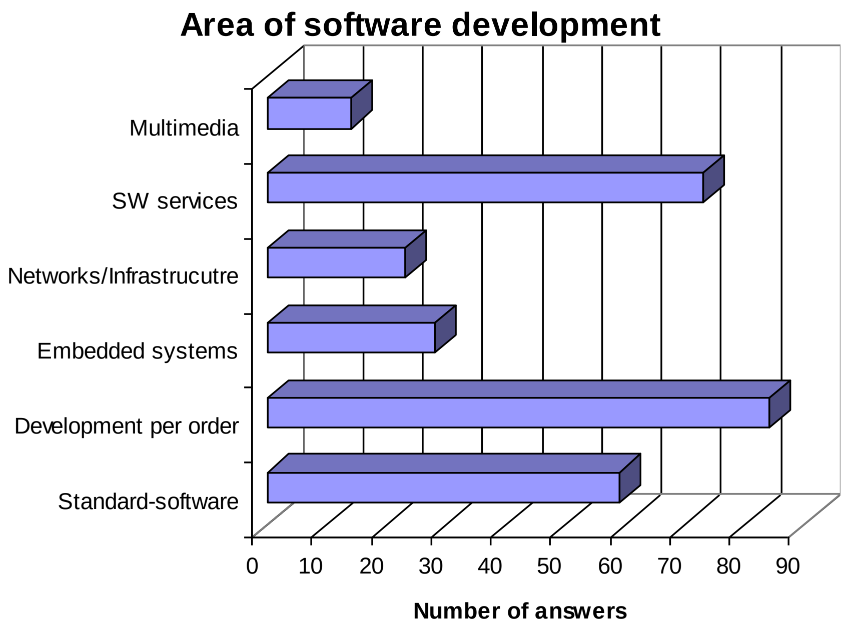


Fig. BK 3: Represented software market areas

Structure of the questionnaire

The respondents were asked to qualify their company with information regarding the size and turnover, the area of software development and the main application area (branch) of their customers. After that they should identify their main business-oriented improvement goals and deduce the relevant technologies and process, which need to be improved in order to reach these business goals. Moreover the respondents should name the aspects they see as biggest barriers to reach their improvement goals and the kind of assistance they are looking for.
In the questionnaire the respondents should “tick” a number form “-3”, which meant total disagreement to “3”, which meant total agreement.
The following figure (Fig. BK 4) demonstrates a part of the questionnaire, regarding the desired assistance. All other aspects were addressed in the same way.

Desired Support **Which support do you wish for your improvement activity**	disagreement			indifferent			agreement	not rele-vant
▪ Demonstration of successful example projects	③	②	①	⓪	①	②	③	○
▪ Feasibility studies / development of prototypes	③	②	①	⓪	①	②	③	○

Fig. BK 4: Example of questionnaire

Goals of improvement activities

All of the possible business goals were rated relatively high. The ratings of all business goals are significantly above zero. Only very few questionnaires contained a negative “tick” (disagreement to a business goal), 4% of the respondents marked “cost reduction” below zero and 2% marked “security of investments” as a disagreement of their business goals. Fig. BK 5 shows graphically the arithmetic average of the most important improvement goals.

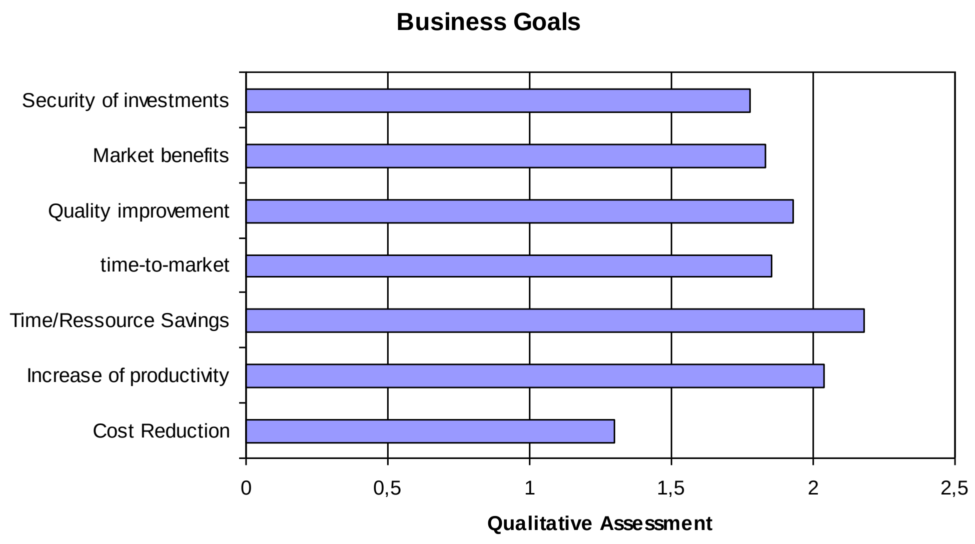


Fig. BK 5: Arithmetic average of the improvement goals

According to the arithmetic average both goals time/resource savings (77% of the respondents demonstrate a large agreement, which means rated with 2 or 3) and productivity improvement (72% demonstrate a large agreement) are regard as being of highest importance. Even cost reduction is significantly above the rating of indifference (average of cost reduction is 1,30), although it is the least important business goal. Many of the respondents kept an eye on the market: 61% of the respondents regard the market factors "time-to-market" and "market benefits" as a highly important business goal. In order to identify the reasons behind this qualitative assessment we conducted some 15 interviews with companies, which completed the questionnaire. It came out that the two highest rated business goals "time/resource savings" and "increase of productivity" are due to the situation on the market. It is relatively easy to get software development contracts. The demand for highly innovative software products and services is very high, especially in the area of embedded systems and e-commerce. Unfortunately it is very hard to take on new employees (due to the great lack of qualified personnel). Therefore especially savings in human resources and increase of productivity is seen as a potential short-term solution to cope with the current situation. Even if every company is always interested to cut costs, this is the least important concern in the actual booming situation (market forecasts predict an annual increase in the turnover of software development of more than 20% [6] for the next few years).

Technological challenges in the next years

Not surprisingly, technological issues, which are necessary to put the „E-Economy" in place, are regarded as the most important technological challenges in the next few years. Almost every software developing company sees an important part of their revenue coming from "e-business" products and services.
The following figure (Fig. BK 6) gives an overview on the calculated arithmetic averages of the technological challenges.

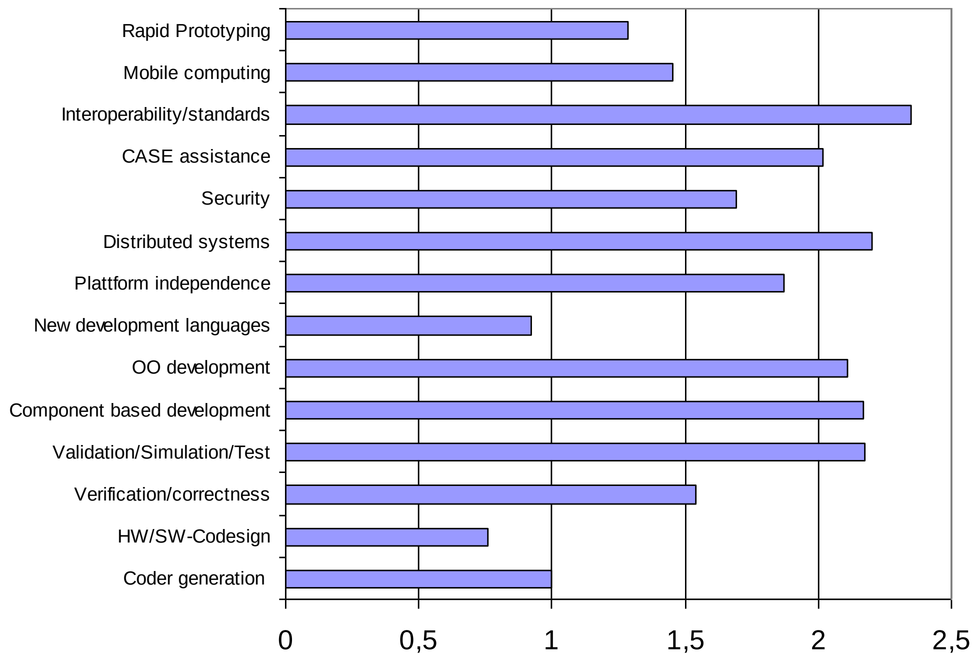


Fig. BK 6: Arithmetic average of the technological challenges

Interoperability is in the average sees as the most important technological challenge within the next few years. 77% of all respondents consider interoperability having a very high importance for their development activities, the arithmetic average is 2,35. Also other technologies which are necessary to deliver products and services for the web-based community are of very high importance for the software developing companies. Distributed systems get an arithmetic average of 2,20 and 82% of all respondents say it is one of the major technological goals to have distributed systems. But also object-oriented development (arithmetic average is 2,10 and 79% say OO has great importance) and component based software engineering (arithmetic average is 2,17 and 77% say it is essential) have a high significance. All these technologies form the technological basis for web-based economy. One of the respondents put it this way: "*If you do not come up with a web-based solution in the next two years, even if it is only for marketing purposes, than just forget your company*". An other remark was "*we have to announce that we always use the newest technology, last year it was XML, this year it is WAP, nobody know what next year is "hip*".

Regarding the low arithmetic averages of all technologies important for embedded systems (like HW/SW Co-design or code-generation) it has to be kept in mind that only some 18% stated that they are involved in the development of embedded systems. The author does not consider this study to be representative for the area of embedded systems.

Targeted process improvements

The process improvement must go hand in hand with technological changes. If we look at the figures presented by the Software Engineering Institute in the Process Maturity Profile of the Software Community 1999 [11] it is clear that a lot of companies are still on very low CMM-level. Therefore it is obvious that the development processes must be dramatically improved. "*It is now obvious the key to profitability and survival is more than just developing a technical solution. It also depends on how well we manage our resources (people, technology, equipment,...), how well we integrate these resources, and the organization's adaptability to change*" [11].
As a matter of fact the arithmetic average of the identified process improvement area is much lower than the arithmetic averages of the technological challenges. Whereas in the technological challenges

the average of the items is between 1,50 and 2,50 it is in the development processes between 0,50 and 1,50. This gives a first impression on the stated importance of technologies versus processes.
The following figure (Fig. BK 7) gives an overview on the calculated arithmetic averages of the identified process improvement areas.

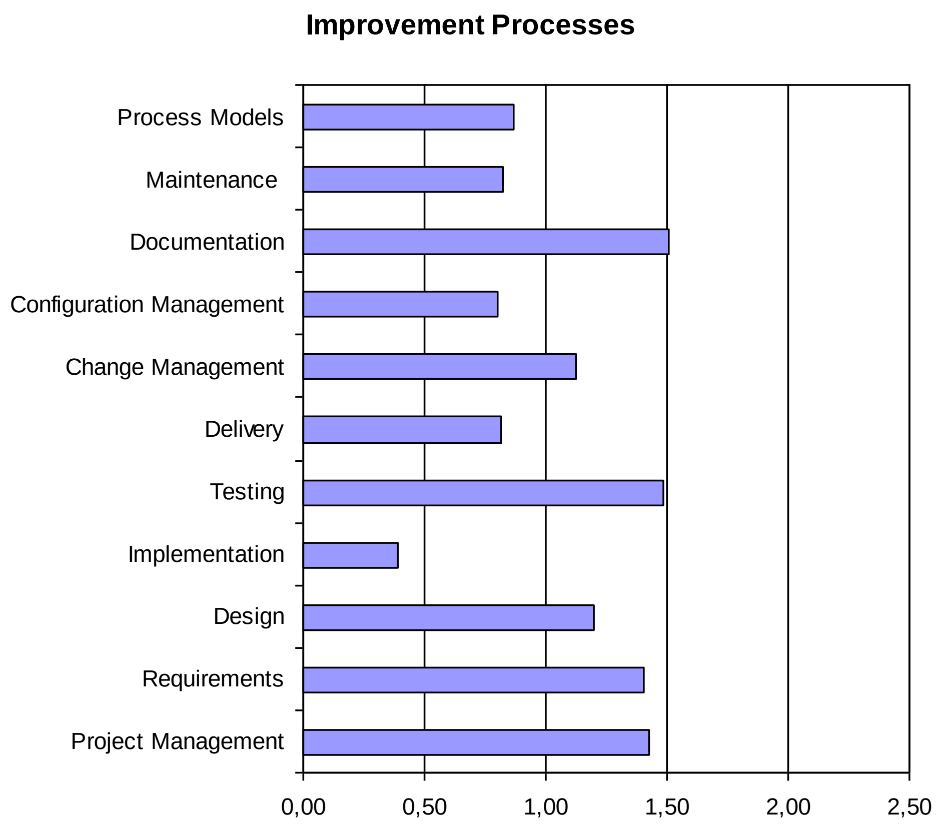


Fig. BK 7: Arithmetic average of the identified process improvement areas

Especially documentation (arithmetic average is 1,51), testing (arithmetic average is 1,48), project management (arithmetic average is 1,43) and requirements analysis (arithmetic average is 1,41) are seen as the most important process-oriented improvements aspects.
These aspects remain the same for several years, since the early beginnings of software process improvement items like project management or configuration management are on the top list of improvement activities. This is based on the fact, that in the last years a lot of new companies, which are now active in new areas like e-commerce or multi-media, were established. The new companies still have the problems established ones had ten years ago. This leads to the question, if our educational systems are adequate, or if important practical aspects are not adequately addressed.
Within the interviews we got the impression that the "land-rush" for technology is so dominant that all process-oriented issues are totally forgotten. One of the interviewees stated it as follows: "*I do not care one iota about the development processes, we have to cope with the technological challenges. I know that we have problems with some processes like configuration management or requirements gathering, but I have no time to improve that, We concentrate now on the technologies and in a few years on the processes*". This leaves the problem, if they will ever have time to cope with process improvement, if there is not always a new technology, which has to be introduced into the company.

Identified Barriers

In order to support software-developing companies in their SPI-activities it is necessary to know the barriers, which hinder the improvement of the processes and used technologies. The study intends to identify the main obstacles. The following figure (Fig. BK 8) gives an overview on the calculated arithmetic averages of the identified barriers.

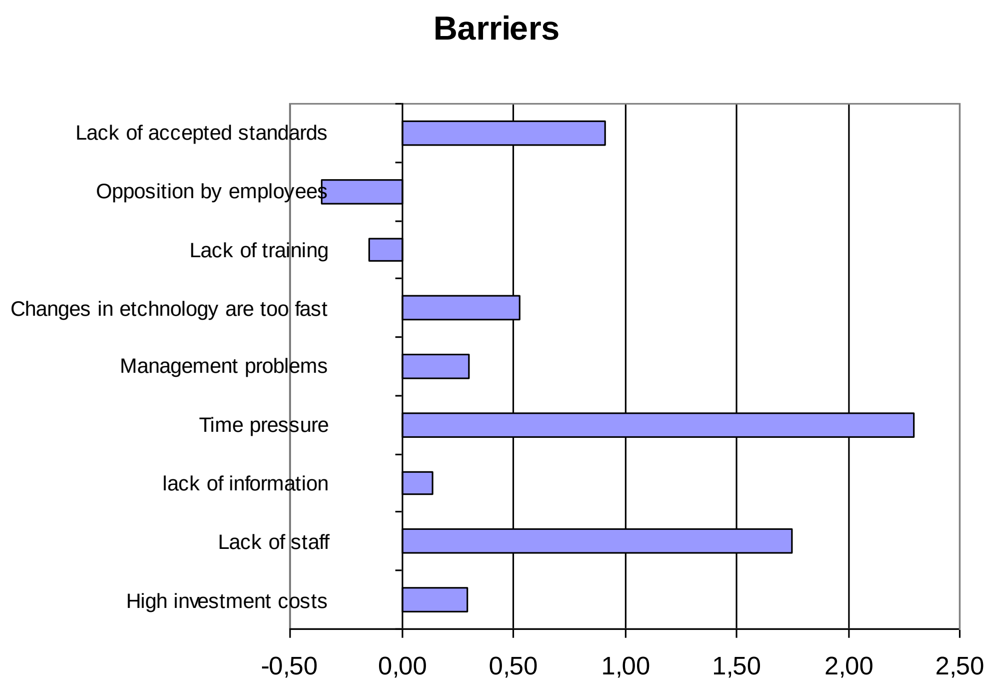


Fig. BK 8: Arithmetic average of identified barriers

Time pressure (70% of the respondents mention time pressure as a very important obstacle) and lack of adequately educated staff (60% of the respondents indicate lack of staff as a major problem) are seen as the major barriers to successful improvement of technologies and development processes. Both items correlate very highly with each other, since time pressure and lack of staff are closely related. This fact fits perfectly to the identified major business goals, time/resource-savings and increase in productivity.
The other very important barrier is the lack of accepted standards. In the interviews the respondents complained about the lack of widespread standards, which is currently a big problem in the e-commerce area. This problem is connected with the desire to reach a good interoperability between solutions (e.g. XML might be a possible solution). A lot of companies (especially established onea) await with the implementation of new technology until universally valid standards are defined. But very often it does not last long till the market is insecure again by new formats and technologies (land-rush of technologies). Internal opposition on staff, lack of good training, lack of information and too high costs of investments are not considered as big obstacles for the introduction of better processes or new innovative technologies. A fast change in technology can face companies with problems. This is not deemed as the biggest difficulty, but a lot of respondents name it as a sort of background problem. In the interviews some respondents showed qualms in respect of the technologies they must patch because of customer compulsion.

Leverages and desired assistance

Most companies need assistance to reach their stated business goals. External assistance of not "pre-occupied" experts is one of the most quoted success factors in change of technology and improvement of processes. This also evident by the recent success of big consulting companies. The following figure (Fig. BK 9) gives an overview on the reported arithmetic average of the desired assistance.

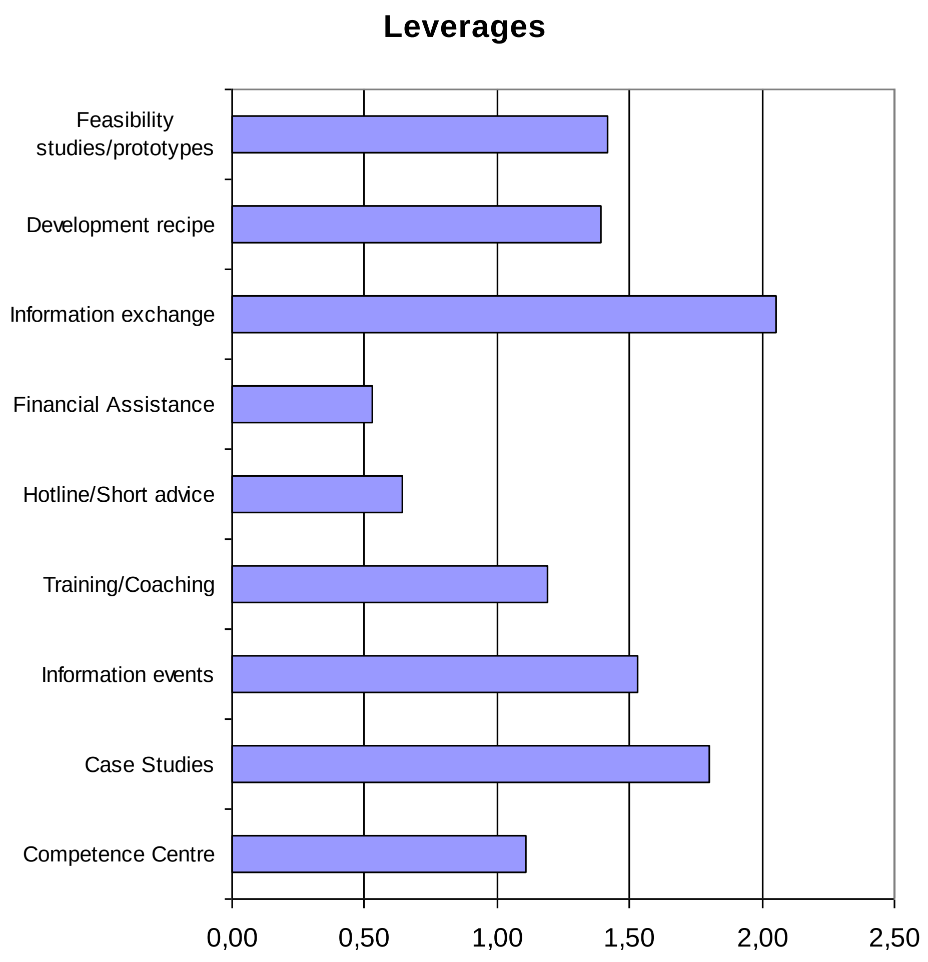


Fig. BK 9: Arithmetic average of the desired assistance

It is very interesting, yet not very surprising, that financial support has received the lowest arithmetic average of the identified assistance areas. This correlates highly with the recognised business goals (cost reduction is seen as the least desirable business goal).
For most of the respondents the exchange of information with experts and colleagues on events is the most important sort of support. Together with information events this is an essential way of planning the future growth of the company. The respondents also postulate the widespread sort of technology transfer by feasibility studies and pilot projects. This strategy is followed by the European Commission in its „Best Practice Projects". In the internet (URL http://www.esi.es/ESSI/Reports/All/24151/Download.html) you can find several case studies of the SISSI Initiative, which cover almost all improvement areas in the process area and the technology area. These studies describe the strategy and the solved problems, the used processes, the chosen technologies and the realised business benefits.
Though the topics training/coaching and the existence of information and consulting centres have not the highest averages, there is a big majority which would like to use such facilities and offers. 95% of all respondents mark the assistance through established competence centres with a mark of zero or higher, and still 84% of all respondents do this for training/coaching in relevant areas.
In the interviews some software experts stated "*we need to go to technology events, there we can see the newest technologies already introduced into a prototype. This gives us an idea about what can be done*". Another one said "*technological feasibility is the most important area of information. It helps us most, if we can have a discussion with somebody who did it.*"

Conclusions

"*The Roman bridges of antiquity were very inefficient structures. By modern standards, they used too much stone, and as a result, far too much labour to build. Over the years we have learned to build bridges more efficiently, using fewer materials and less labour to perform the same task.*" quoted from Tom Clancy (The Sum of All Fears).
Software development is a relatively new field in science and practice. There are huge gaps between the state of the art and the targeted goal. We can only hope to build modern software efficiently (just like bridges, each major step towards improvement be it technological or process oriented must be investigated, studied, reported and shared) if we improve both, the technology and the processes. It makes no sense to concentrate only on either one [1].

At the moment all business goals are greatly influenced by the lack of staff and the emergence of new products and services in the e-commerce field.
Very interesting is the difference between the technological goals, which are highly innovative and most of them rather new and the "old" process-oriented problems, which remains the same all the time.
The technologies, which should be introduced into the companies within the next few years, are highly innovative (due to demand of distributed, web-based solutions). The processes, which need improvement, remain the same, since first SPI activities started. Documentation, project management and testing, which are often seen as bureaucratic activities within the software development life cycle still need to be improved. The most important barriers in order to reach the targeted business goals are the lack of staff and time pressure within the projects (related to the current situation with several hundred thousand software-experts missing in Europe) and the lack of accepted standards (especially for interoperability). The companies appreciate, if multipliers or competence centres organise information exchange meetings. For the demonstration of feasibility, pilot projects and demonstrators are seen as very helpful. At the moment Money is not the problem, it is the least desired business goal to reduce costs and it is the least quoted assistance measure to receive financial support.
Notwithstanding, this study is hardly in-depth enough to provide a real solution to such a daunting problem as the current low level of software development. In order to reach high quality of software products and services, to keep schedules and cost calculations, a lot of effort is necessary. Especially the assisting organisations should concentrate more on the business goals of their customers and not on their consulting approaches. Since almost all technology matters are attributed with higher importance than the process issues, it seems to be necessary to consider appropriate activities.

Here ends this report, it should serve as a reminder for discussion and leaves us with the questions:

- Is it sufficient to concentrate only on processes? Is it necessary to combine technology and process aspects in an improvement project?
- How can we attract more software developing companies to improve their processes?
- Are new approaches or assessment methodologies (which combine processes and technology) necessary?
- Is it feasible to go for both a technology improvement and a process improvement in one SPI-project?

References

[1] Watts, Humphrey S.: Doing Disciplined Work . SEI 1994
[2] Koch, Guenther R.: Process assessment: the "BOOTSTRAP" approach. 1994
[3] Rombach, Dieter: Software Qualität geht alle an. Handelsblatt Ausgabe 49, 1998
[4] Rombach, Dieter: Softwareprozessverbesserung: Strategische Notwendigkeit für alle Branchen in SQM (Software-Qualitätsmanagement) „Made in

Germany“ – Nützlich oder strategisch notwendig- SQS Gesellschaft für Software-Qualitätssicherung mbH Köln, 1998
[5] The Standish Group: CHAOS-Report. The Standish Group, Report, 1998
[6] Sentry Technology Group: 1997 Software Market Survey - Establishing value-based supplier differentiation. http://www.softwaremag.com/ research_reports/soft_mkt/ Download 06.07.2000
[7] Business Software Alliance: Forecasting a robust future, Report 1998
[8] European Software Institue: Process Improvement - The Business Case. http://www.esi.es/Brochure/English/br2.html Download: 03.07.2000
[9] Rementeria, Santiago: EUROPEAN SOFTWARE INSTITUTE in SOFTWARE PROCESS-Improvement and Practice - Vol.1, Issue 2 1995
[10] Wiegers, Karl E.: Process Improvement that Works. In Software Development. 1999
[11] Carnegie Mellon University - Software Engineering Institute: Process Maturity Profile of the Software Community 1999 Year End Update

Appendix 1 - Curriculum Vitae author

Bernhard Kölmel is currently head of the divisions “International Department” and “Technology Transfer” at FZI Forschungszentrum Informatik Karlsruhe. FZI is an interdisciplinary technology transfer institute, which is specialised in industrial take-up projects of highly innovative information and communication technologies (ICT).
In this respect he is responsible for the management of all public R&D projects and the contacts to national and international industry.
Mr. Kölmel is project leader of several national and international R&D and Technology Transfer projects, among them IMPROVE software: the ESPINODEs (European Software Process Improvement Node), the SME COMPETENCE CENTRE for Electronic Commerce, Virtual Organisations and Knowledge Management, the project VIRTUAL OFFICE which provides knowledge intensive services for SMEs etc.
Mr. Kölmel is since June 1999 acting head of the division “Business Process Engineering and Management”, where he is responsible for the establishment and the growth of the division.
Mr. Kölmel is advisor in European ICT matters (esp. IST programme) to ministry for economic affairs of the federal state of Baden-Württemberg. He is involved in the set-up of public funded initiatives in the area of Software Process Improvement, Business Software, Embedded Systems and Virtual Organisation for various ministries in Germany.
Mr. Kölmel is also evaluator and reviewer for various projects and programmes of the European Commission.

Appendix 2 – description of company

FZI Forschungszentrum Informatik Karlsruhe is an interdisciplinary technology transfer institute, which is specialised in industrial take-up projects of highly innovative information and communication technologies (ICT).
FZI was established to bridge the gap between Universities and Industry. FZI helps its customers with their investment policies by supplying them with critical information. In feasibility studies FZI shows hether and how an innovative technology might be useful. Comparative case studies help to choose among business and engineering methods and techniques, and market analyses put a customers in a position to select software and hardware tools. The following figure illustrates the approach of FZI to technology transfer.

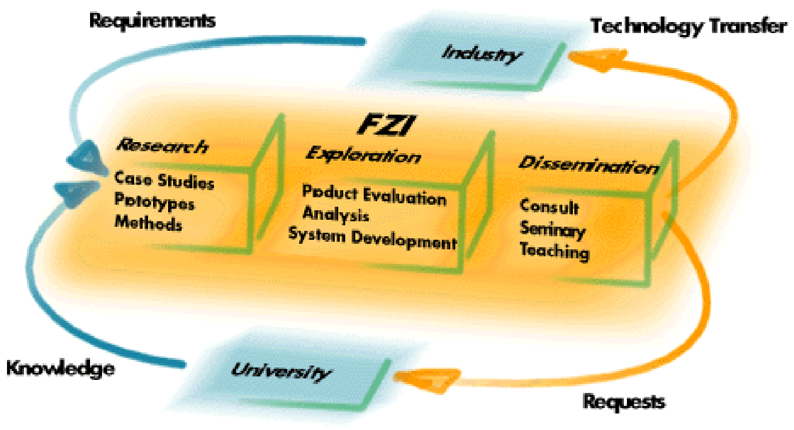


Fig. BK 10: FZI – ways of technology transfer

FZI's Expertise for Tomorrow's Solutions
With our core competence in software and hardware systems, particularly for supporting distributed applications, FZI occupies a central position in modern information and communication technology. FZI focus its activities on a number of critical areas.

- Software technology
- Micro Electronic Systems
- Distributed automation systems
- Distributed engineering
- Distributed information systems
- support and on component-based workflow architectures.
- Data communications for distributed and multimedia applications
- Mobility services
- Service robotics

An Interview Based Evaluation of a Process Change Proposal

Martin Höst
Department of Communication Systems, Lund University P.O. Box 118, SE-221 00 LUND, Sweden
Martin.Host@telecom.lth.se

Tomas Berling
Ericsson Microwave Systems AB
SE-432 84 Mölndal, Sweden
Tomas.Berling@emw.ericsson.se

Introduction

Process improvement means changing the currently used processes. It is, however, almost never certain that a proposed change actually will result in an improvement. Even if available methods for process improvement [10] help in deciding what changes to choose, they do not say exactly how changes should be made. Instead of an improvement, there may be no effect at all, or worse, it may result in that processes that previously were working as they should, are affected negatively. It is also possible that the process will be improved with respect to one aspect, but it will work worse than before with respect to another aspect. Hence, evaluation of improvement proposals is an important task, which should be started as early as possible in the improvement process.

Evaluation can be performed in a number of different ways, such as in controlled experiments, in case studies, and in pilot projects [4, 8]. In this study an evaluation is performed mainly through interviews with the staff before any of these approaches are adopted. This paper presents a study where a process change proposal in an industrial organisation is analysed. The process change proposal consists of introducing a new review method when verification specifications are reviewed.

One advantage of performing a study this early before experiments and case studies is that information can be gained that can be used in the experiments and case studies. For example, if the interviews show that the change proposal will not result in any improvement, this means that the change proposal can be reformulated without that any effort has to be spent on actually using the first proposal in the organisation.

The investigation is mainly performed as a survey [9] through interviews with staff in the organisation. However, the change proposal has been introduced and used after the interviews. Therefore, some evaluations are also made based on data from a limited number of reviews.

The organisation and its processes

The presented study is performed at Ericsson Microwave Systems AB in Sweden. This paper is concerned with the processes dealing with verification specifications and verification. The description is therefore focused on these areas.

Generally, verification is performed in three steps in the organization:

- Sub-system test, where parts of the system are tested.
- System integration, where sub-systems are merged to a system. The evaluation presented in this paper deals with this step.
- System verification, where a complete system is tested.

The organization consists of three units with responsibilities related to verification:

- System design unit, which is responsible for the software design and the requirements specification. The unit develops new requirements specifications with the previous as a basis. The system is extended, but many functions can be reused.
- Software design unit, which is responsible for implementing the design. The unit develops new code with the previous system as a basis. The unit also performs sub-system tests.
- Verification unit, which is responsible for sub-system test, system integration, and system verification and validation. The unit develops test specifications and performs the tests and analysis. It is an independent unit.

The organization develops radar systems. The systems handle broad-band information with hard real-time requirements. Several projects are handled within the same organization.

In the organization, improvements and changes are performed continuously. A process improvement program is ongoing, and continuous improvements are performed in a number of areas. This study is performed for a change, which is one part of the improvement program. The change is described in the next section.

Change proposal in this study

The study is performed in the context of system verification and integration of a modified system, see Figure MHTB.1.

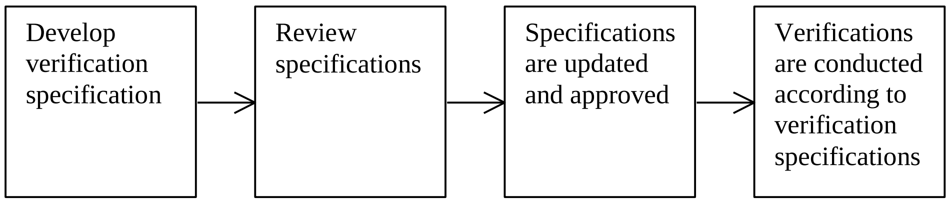

Develop verification specification
Review specifications
Specifications are updated and approved
Verifications are conducted according to verification specifications

Figure MHTB.1: System verification and integration.

The analyzed process change proposal consists of the introduction of a New Review Method (NRM) for review of verification specifications. Previously, the specifications were reviewed, but the review process was not as structured as NRM.

NRM is a formal inspection method conducted by 3-6 selected persons who in a formalized and standardized way examine a document (or other object) to find defects. The method is defined to avoid discussions about, for example, the purpose, conduction, and responsibilities of the document.

When reviews performed according to the method are initiated, a trained moderator, who plans and leads the work, is assigned to the work and reviewers are decided. Then the document to be reviewed is distributed to the reviewers together with information about the review and roles. After individual reviews a review meeting is held. The results are presented in a protocol, which is presented to the author and the responsible manager as a basis for a decision concerning either rework or acceptance. The method also includes procedures for collection of metrics for process improvement work. NRM includes measurements of the review process and the specifications, both process and product attributes.

In the organization NRM is used for review of a number of other types of documents too. The main objectives of introducing NRM are to improve the quality of the reviews and to increase the efficiency when reviewing the specifications. More faults would be found at an early stage resulting in higher quality of the developed systems and the systems being developed in a shorter time. The objective is also to increase the possibility of improving the development of specifications and the review process. Additional objectives are to increase the knowledge of the system, by letting more people read the documents more thorough, and to get better indications of the quality of the document at the meeting.

Evaluation method

The investigation is based on interviews with people in the organization, and follows the process impact analysis method, which is described in general in this section. The method is described in more detail in [5].

The objective of the impact analysis method is to provide a means for performing early evaluation of improvement proposals. This can be a step to perform before a change is analysed in, for example, an experiment or a case study. The impact analysis method describes three steps that can be carried out in order to estimate the effects a change proposal:

- Estimation on sub-processes: For every affected sub-process, people are inter-viewed concerning the effects of the change proposal on that sub-process. The result of this step is one prediction of the effects of the change proposal for every sub-process. This step is further described below.
- Overall prediction: Since the objective of the method is not only to estimate the effects on sub-processes, it is necessary to obtain a prediction of the effect on the overall process. This is the objective of this step. The overall prediction can either be performed mathematically based on the estimations for the sub-processes, or more informally, by comparing the effects on sub-processes and discussing the effects on the overall process. In the investigation presented in this paper the latter, more informal approach has been chosen.
- Presentation of results: When the effect of a change proposal has been estimated it must be presented both to the interviewees and to management.

Estimation on sub-processes

The study presented in this paper is to a large extent based on interviews according to the process impact analysis method. Therefore, the first step of the method is further described in this section. How to perform interviews in the impact analysis method is also described and discussed in [6, 7], where a number of alternative ways of asking questions are described and compared.

For every sub-process, a number of persons are interviewed. They can for example be given the question: ‘If the process is changed, how large is the relative effort of the sub-process compared to before the change?’. This questions could be answered in a number of different ways, such as with a point estimate, an estimation according to a rectangular distribution (i.e., a smallest possible value and a largest possible value), or an estimation according to a triangular distribution (i.e., a smallest possible value, a most likely value, and a largest possible value). There are also different ways of aggregating the individual estimates when an estimation for the sub-process is derived. In [6, 7] a number of different alternatives are compared, and a technique for estimation of the effects on the sub-process based on individual estimates is recommended. This technique is used in the evaluation presented in this paper. The technique is based on that every person performs estimations according to triangular distributions (i.e., a smallest possible value, a most likely value, and a largest possible value are given). This means that for every interviewed person, a distribution is obtained. An aggregated distribution can then be derived as the mean of the individual distributions. This distribution can be used to estimate, for example, a mean value and a variance for the sub-process. The variance, or another measure of dispersion, could be used as a measure of uncertainty. In this paper a normalised variance, c^2, calculated as the variance divided by the square mean is used as a measure of uncertainty.

Here, analytical estimations are performed as described in [6], but estimations could also be based on simulations (see for example [3]).

The evaluation described in this paper is based on the impact analysis method, as described above. The details of the evaluation, such as the choice of question topics, etc., are based on the objectives of the evaluation. In the Goal Question Metric paradigm [1], often used together with the Experience Factory concept [2], it is stressed that measurements should be relevant, and based on improvement goals. This is also important here. It is important that the interview questions represent the improvement objectives of the organisation. Therefore, the topics for the interview question and the modelling of the process was performed in co-operation with the organisation.

The main objective of the investigation is to investigate the effects of introducing NRM. The investigation focuses on reviews of verification specifications and analyses the effects of the reviews on:

- Quality of specifications: The objective is to evaluate if the introduction of NRM improves the quality of the specifications when the verification is started. The quality is measured through the number of faults that are found in reviews.
- Quality of verified products: The objective is to evaluate if the introduction of NRM increases the number of faults found in the product during verification. The quality is measured through the number of faults in the system that are found in verification when the reviewed specifications are used.
- Effort: The objective is to evaluate if the introduction of NRM affects the totally required effort. This includes effort required in reviews, effort required for development of specifications, and effort required in verification.

No specific training has been performed concerning how reviews were performed before the change, or concerning how reviews are conducted according to NRM. It is assumed that most persons in the organisation are familiar with how reviews were performed before NRM was introduced, and that people are familiar with NRM from reviews of other types of documents, such as verification specifications for sub-system test. It is not necessary that people have experience from using NRM in reviews of verification specification for system verification. The experience of the interviewed people is, however, assessed as part of the investigation.

Validity of the results

Since this study is based on interviews, the result is uncertain. It is of course, impossible to know exactly what the results of introducing a change is before it is introduced. The fact that the investigation is based on subjective opinions is probably the largest origin of uncertainty in this study, but on the other hand it allows for a first estimation of the impact of the proposed process change.

Some other sources of uncertainty in a study like this are

- Modelling: When a study like this is performed, it is important that the changed process and the original process are modelled as a basis for developing questions for the interviews. There is always a risk that an interviewed person misunderstands a question and therefore answers it in the wrong way. This risk can be minimised by carefully modelling the processes.
- Previous changes: It is not common that the processes have been unchanged before the change. In many cases the processes have been continuously improved. This means that it is important to try to sort out the effects of the investigated change from the effects of previously conducted changes.

The intention has been to make the uncertainties as small as possible in this study. However, there is always a risk of misunderstandings and there is always a risk that differences are due to previously conducted changes as well as the investigated change proposals.

The interviews

Since all the interviewed persons do not have the same experience, all interviews are started with a number of characterisation questions. The answers to these questions are used to measure the experience. If a person’s experience is not considered to be sufficient to answer a question the answer is not used in the analysis. This means that all interview answers are not used when the answers to every interview question are analysed.

All persons are classified according to a number of criteria, which have been defined for the study. Examples of criteria are if a person has carried out reviews with the NRM, or if a person has been involved in verification in earlier work.

Main interview questions

After the initial questions concerning experience have been asked, questions concerning the effect of the review method on different sub-processes are asked. Questions are asked concerning the two cases that NRM is not used, i.e., concerning the situation before NRM was introduced, and concerning the case that NRM is used, i.e. the new situation. The questions, divided into topics, are:

Preparation effort: Questions are asked with respect to the effort required to develop a verification specification before the review. Every interviewed person is asked an initial question:
Qpre1: Does the review method affect the effort before the review?
If the answer is yes, two additional questions are asked:
Q1: What is the required effort before the review if NRM is not used?
Q2: What is the required effort before the review if NRM is used?

Review effort: The review effort denotes the effort required per person to review the verification specification. It includes both preparation and meeting time. Two questions are asked:
Q3: What is the required review effort if NRM is not used?
Q4: What is the required review effort if NRM is used?

Update effort: The update effort denotes the effort required to update the verification specification after the review. Two questions are asked:
Q5: What is the required update effort if NRM is not used?
Q6: What is the required update effort if NRM is used?

Verification effort: The verification effort denotes the effort required to verify the system according to the verification specification. Two questions are asked:
Q7: What is the required verification effort if NRM is not used?
Q8: What is the required verification effort if NRM is used?

Pre-existing number of faults: Questions are asked with respect to the number of faults that exist in the verification specifications before the review. Every interviewed person is asked an initial question:
Qpre2: Does the review method affect the number of faults in the verification specification before the review?
If the answer is yes, one additional question is asked:
Q9: What is the relative number of faults in the verification specification if NRM is used compared to if NRM is not used?

Number of faults in review: This measure denotes the number of faults that are found in review. Two questions are asked:
Q10: How many faults are found in verification specifications during review if NRM is not used?
Q11: How many faults are found in verification specifications during review if NRM is used?

Number of faults in specification in verification: This measure denotes the number of faults that are found in the verification specification during verification. Two questions are asked:
Q12: How many faults are found in verification specifications during verification if NRM is not used?
Q13: How many faults are found in verification specifications during verification if NRM is used?

Relative number of faults earlier with NRM: This denotes the relative number of faults (in verification specifications) that can be found already in review instead of in verification. One question is asked:
Q14: How many of the faults in the verification specification that normally are found in verification could be found in reviews instead if NRM is used?

Number of system faults in verification: This measure denotes the number of system faults that are found in the system during verification. Two questions are asked:
Q15: How many faults are found in the system during verification if NRM is not used for review of verification specifications?

Q16: How many faults are found in the system during verification if NRM is used for review of verification specifications?

Answers to the questions

The interviewed persons are asked to answer questions Q1-Q16 according to triangular distributions as described above.

For the question topics where there are two questions, i.e., one question concerning reviews according to NRM and one not according to NRM, the person could choose to answer relatively instead. For example, questions Q5 and Q6 deal with the update effort if NRM is not used and if NRM is used. In this case, it is possible for the interviewee to answer (according to a triangular distribution) what the relative update effort would be if NRM is used compared to if NRM is not used. For the persons who give a relative estimate, an absolute estimation for the case when NRM is used (Q6 in the example) is approximated. The approximation can be done in a number of ways, here the following procedure is adopted:

- Determine mean, m_r, and variance v_r, of the distribution given as a relative estimate. Determine mean, m_n, and variance, v_n, of the result of the estimation with respect to the question concerning the case when NRM is not used (Q5 in the example). That is, m_n and v_n are based on the answers from persons who provided absolute answers.
- Approximate mean, m, and variance, v, as the Gauss approximation of the mean and variance of the product of two independent stochastic variables with mean, m_r, and variance v_r, and mean, m_n, and variance, v_n.
- Use a symmetric triangular distribution with mean m, and variance v as an approximation of an absolute estimate for that person.

Since not all people have experience from all phases in the organization, decisions must be made concerning which answers to use when estimations are performed. The defined criteria described above were used to determine which answers to use.

Results from the interviews

Before reviews

The objective of questions Q1, Q2, and Q9 is to assess if there is any difference with respect to the work with the verification specifications before the review if NRM is used. It may, for example, be the case that more effort is devoted to the specifications if they are to be reviewed with NRM.

Of the persons that have the experience that fulfil the criteria, 5 persons thought that the effort before the reviews is affected by the review technique (Qpre1). One of these persons chose to give a relative answer. The other 4 persons were able to estimate an absolute value for how much effort that is required before the review if NRM is not used (Q1). These 4 answers are used in the estimation for the case that NRM is not used, and the result is a mean value of 125 hours and a c^2 of 0.75.

For the case that NRM is used (Q2), the estimation is 143 hours with a c^2 of 0.78. This estimate is, however, only based on the opinions of the people who said that the review method affects the effort before the review. If estimates are used also from persons who said that the review method does not affect the effort before the review, the difference becomes smaller. For example, if the estimates from these people are approximated by a symmetric triangular distribution with the mean value and standard deviation according to the estimation for the case that NRM is not used (i.e., Q1), the mean value would be 131 hours.

That is, the result is uncertain (e.g., c^2=0.75 for Q1) and the difference is small.

For Q9 (the relative number of faults in the verification specification if NRM is used compared to if NRM is not used) 5 persons thought that the number of faults in the verification specification before the review is affected (Qpre2) by the review method. The estimation results in a mean value of 0.54 and a c^2 of 0.23. This indicates a large difference between the old and new situation, but the estimation does not consider the opinions of the experts who said that there is no difference. If those persons' estimates are approximated by a point estimate of 1, the mean value of the estimation would be 0.83. This is still a difference that is worth considering, although it is smaller.

The specifications seem to be of higher quality if they are to be reviewed with NRM. The difference with respect to effort before the review is, however, small.

Effort

The estimated effort required in phases directly affected by NRM is shown in Table MHTB.1.

Table MHTB.1: Effort in different phases.

Question	# answers	mean	c^2
Q3, review effort per person without NRM	14	8.73	0.48
Q4, review effort per person with NRM	14	6.32	0.36
Q5, update effort without NRM	9	17.5	0.46
Q6, update effort with NRM	9	18.7	0.82
Q7, Verification effort without NRM	8	258	0.60
Q8, Verification effort with NRM	11	247	0.78

The review effort (preparation + meeting) seems to be smaller if NRM is used. This is probably because the meetings are more structured. No special questions have been asked concerning the preparation for the reviews (i.e. the preparation time before the meeting), but it is possible that this time is slightly longer if NRM is used, since there are higher requirements on preparation with NRM.

There is only a very small difference between the effort required for updating the specification if NRM is not used compared to if NRM is used. A little more effort may be required if NRM is used. If there is a difference, one explanation to this could be that more serious faults are found if NRM is used, since the simpler faults are removed already before the review. The difference is, however, very small and it is not possible to conclude that there really is a difference.

The differences obtained with respect to the effort required for verification are very small.

Number of faults

The estimated number of faults found in different phases is shown in Table MHTB.2.

Table MHTB.2: Number of faults found in different phases.

Question	# answers	mean	c^2
Q10: Number of faults in review without NRM	14	36.7	0.71
Q11: Number of faults in review with NRM	13	36.3	0.53
Q12: Number of faults in verif. without NRM	12	10.8	0.32
Q13: Number of faults in verif. with NRM	10	9.42	0.36
Q15: Number of system faults without NRM	9	17.0	0.88
Q16: Number of system faults with NRM	13	16.9	0.68

The analysis of Q10 and Q11 indicates that about the same number of faults are found in reviews of verification specifications if NRM is used as if NRM is not used.

Even if the difference is small the analysis according to Q12 and Q13 indicates that some fewer faults will be found in the specification during verification if NRM is. A decrease is considered positive for the new method. The difference is, however, small. The analysis according to Q14 indicates that about 25% of the faults that previously were found in the specifications during verification now will be found already in review if NRM is used. The results are, however, very uncertain ($c^2 = 1.3$).

The analysis according to Q15 and Q16 indicates that there is no difference between the number of faults that are found in the system in verification if NRM is used and if NRM is not used.

Results from reviews

Reviews of three different verification specifications have been performed with NRM. The results of the reviews are summarised in Table MHTB.3.

Table MHTB.3: Summary of review data

Review	Review effort/person (h)	# reviewers	# faults in review
1	25.3	6	30
2	10.0	5	61
3	12.5	6	36
mean value	15.9		42.3
from interviews	5*6.32 = 31.6 (based on Q4)	(5)	36.3 (based on Q11)

This shows that the estimation with respect to review effort was too high. That is, the interviewed people anticipated that reviews would require more time than actually was required. The estimation of the number of faults that are found in reviews was somewhat more accurate. No estimation technique can give precise and correct estimations every time it is used. Estimation errors will always occur. In cases like this when the quantitative experience is limited, the measured values are uncertain. The review data are based on three measurements, which is not enough to form a reliable baseline. This should be noticed when the estimated values are compared to the measured.

There are only uncertain measurements available from reviews where NRM was not used. Therefore, it is not possible to compare the estimations with respect to this case when NRM is not used. However, the interviews indicate an improvement from 8.73*5=43.6 (Q3) to 31.6 hours. The measurements indicate an even larger improvement.

When the actual results are compared to the estimated results it can be concluded that the estimates in this case are of the same magnitude as the real results, but there are some estimation errors. This must be considered when conclusions are drawn from the estimations.

Conclusions

This paper shows that interviews with people in an organisation may be a feasible way of analysing a process change proposal with limited effort.

The results of the interviews indicate that there are fewer faults in the verification specifications before the reviews if NRM is used. An explanation to this could be that the requirements on the document are set higher by the author if it is to be reviewed with NRM. The difference with respect to effort before the review is very small.

According to the interview results, less effort is required per person in the reviews if NRM is used, but about the same number of faults are found during the reviews.

Concerning the verification phase the results are uncertain, and the differences between if NRM is used and if NRM is not used are small. Even if the difference is small, the analysis also indicates that some fewer faults will be found in the specifications during verification if NRM is used.

The reason that the differences are small may be related to that the results of reviews were more uncertain when NRM were not used. Good reviews were performed also before NRM was introduced, but since the review process was not that defined, the quality of the reviews was not that stable. The general view is that the reviews that were performed for test specifications were of high quality even if NRM was not used

Acknowledgement

This work was partly funded by The Swedish National Board for Industrial and Technical Development (NUTEK), grants 1K1P-97-09673, 1K1P-97-09690.

References

[1] V. Basili, G. Caldiera, D. Rombach, 'The Goal Question Metric Approach', In Encyclopedia of Software Engineering, editor J. Marciniak, John Wiley and Sons, New York, pp. 528-532, 1994.

[2] V. Basili, G. Caldiera, D. Rombach, 'Experience Factory', In Encyclopedia of Software Engineering, editor J. Marciniak, John Wiley and Sons, New York, pp. 469-476, 1994.

[3] L. Briand, K. El Emam, F. Bomarius, 'COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment', Proceedings, 20th International Conference on Software Engineering, pp. 390-399, 1998.

[4] N. Fenton, S.L. Pfleeger, 'Software Metrics - A Rigorous & Practical Approach', Second Edition, International Thomson Computer Press, London, UK, 1996.

[5] M. Höst, C. Wohlin, 'Impact Analysis of Process Change Proposals', In Software Quality, editors by M. Ross, C. Brebbia, G. Staples, J. Stapleton, Computational Mechanical Publications, Southampton, UK, pp. 311-323, 1995.

[6] M. Höst, C. Wohlin, 'A Subjective Effort Estimation Experiment', Information and Software Technology, Vol. 39, No. 11, pp. 755-762, 1997.

[7] M. Höst, C. Wohlin, 'An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimates', Proceedings, 20th International Conference on Software Engineering, pp. 332-339, 1998.

[8] S. Pfleeger, 'Experimental Design and Analysis in Software Engineering', Part 1-5. ACM Sigsoft, Software Engineering Notes, Vol. 19, No. 4, pp. 16-20, Vol. 20, No. 1, pp. 22-26, Vol. 20, No. 2, pp. 14-16, Vol. 20, No. 3, pp. 13-15, Vol. 20, No. 4, pp. 14-17, 1994-1995.

[9] C. Robson, 'Real World Research', Blackwell Publishers, 1993.

[10] H. Thomson, P. Mayher, 'Approaches to Software Process Improvement', Software Process - Improvement and Practice, Vol. 3, pp. 3-17, 1997.

Appendix A, Ericsson Microwave Systems AB

Ericsson Microwave Systems is a wholly owned subsidiary of Ericsson. We specialize in microwave technology and take the full advantage of technical interaction between telecom solutions and defence electronics. The rapid growth in mobile communications for telephony and Internet services puts a strong demand on our products in the area of telecommunication.

The product portfolio comprises digital radio links for wireless transmission solutions and the MINI-LINK(tm) family demonstrates a world market leadership for radio links. Other products represents radio base stations and advanced antenna systems for several standards including the third generation mobile systems.

We are also a world-leading supplier of defence electronics with state of the art products for surveillance and air-defence. Advanced sensor systems as the ERIEYE radar based on microwave technology, represents a new generation of Airborne Early Warning and Control systems. Airborne radar and computers together with ground based radar systems, constitutes a strong product range provided the world market.

We are well positioned to meet the increased demand for information technology and communication networks in military affairs. In close cooperation with other Ericsson units we provide a unique competence to support the strategic change and technology shift that is denominated RMA, Revolution in Military Affairs.

Our combined expertise and technical interaction create valuable synergetic effects for our customers.

Appendix B, About the authors

Dr. Martin Höst is an assistant professor in the Software Engineering Research Group at the Department of Communication Systems at Lund University. His main research interests include software process improvement, early evaluation of software process change proposals, modelling of flexible software processes, and computer simulation of software development processes.

Mr. Tomas Berling is a project engineer at Ericsson Microwave Systems AB in Mölndal, Sweden and an industrial PhD student in the Software Engineering Research Group at the Department of Communication Systems, Lund University, Sweden. He has an MSc in Electronical Engineering from the University of Chalmers, Sweden. He has eight years of working experience in software engineering at two companies. He is now working with system verification and validation of large and complex real-time software systems. At 50 % of his working time he is doing research in the area of system verification and validation. His research interest is system verification and validation of complex real-time software systems.

Session 4 - SPI and Systems

Session Chair:

John Elliott,

SEC, DERA

INTERWORKS, HOW TO SUPPORT THE REALIZATION OF A VIRTUAL SOFTWARE FACTORY

Felicio MANZO
Alenia Marconi Systems S.p.A.
via Tiburtina 1231, 00131 ROME-ITALY
tel. +39-10-6546-240;
e-mail: fmanzo@aleniasystems.finmeccanica.it

ABSTACT:

INTERWORKS ("Introducing Network TEchnologies and Re-organising current WORKing practices in Software development at AMS-DSN") is a project developed by the Naval Systems Division of AMS (Alenia Marconi Systems SpA), a multinational Company leader in the European Defence market. The project has been partially funded by CEE through ESSI-PIE (Process Improvement Experiment).
INTERWORKS is aimed to re-organize the software development process in order to achieve business goals more successfully. The main target is to support and encourage co-operative working practices along the whole software life cycle. The paper is focused on the SRS (software requirements specifications) development step for a Command and Control (C&C) system, part of a large supply having as customer the Italian Navy.
The project serves two relevant internal initiatives having as objective "business improvement", the BEM (Best Excellence Model) and CMM (Capability maturity Model).
INTERWORKS, that started on July 1998 and finished on March 2000, established a new operational scenario by means of: deep training program (involving about twenty AMS professionals), punctual analysis of the "as is" software process, team-based design of the "to be" process, Factory Information Portal development and experiment implementation, evaluation and dissemination.

1. Introduction

Alenia Marconi Systems/Divisione Sistemi Navali, in the following for short AMS/DSN, is a system Division mainly operating in the field of designing, implementing, selling and maintaining complex equipment and systems, both hardware and software, for ships and naval traffic control. The main customers are the Italian Navy Ministry and the Navy Ministries of many foreign countries.
Because the final product is usually customised and software intensive, the customisation strongly impacts on the software design and development process. In addition, each product (complex equipment or system) is normally developed by a team of people having different skills and located in different places.
Because the final cost is a key factor in market competitiveness, INTERWORKS originates mainly from the understanding that it is compulsory to improve the software development process in order to keep (and possibly increase) the present market share, facing international competitors.
INTERWORKS aims to demonstrate, first of all, that it is possible to improve the efficiency of the software development process, reducing the allocated effort, the global costs and the time-to-market.
This is achievable through the reorganisation of the software development process, making available to the users an environment based on the concept of virtual team and virtual factory, which allows information, data, tools and, generally speaking, advanced technology sharing. Therefore, the experiment focuses on the virtual factory metaphor and process re-engineering methodology, in order to support, by means of the electronic technology, co-operative operations of teams composed of people working for different organisations and based in different places, overcoming the time and space barriers.

AMS top management has sponsored INTERWORKS. The Naval Division General Manager assigned the project to the proper Business Improvement & Quality Assurance Director: The Director committed the project to the Process & Information Technology Unit. The Unit manager assumed the project management and staffed the project with two senior engineers, working in the Process and the IT areas respectively. The System Engineering Unit, responsible for the C&C software development, selected as baseline case, dedicated one senior engineer to the project, which co-ordinates the internal engineers
considering the various contributes required by the experiment. The QA Unit and the GSI (Information Systems Management) Central Unit designated representatives to follow the project development in order to evaluate procedural changes and facilitate info-structure usage respectively.
The BI&QC Director authorised the financing from a dedicated account as part of "Business Improvement Internal Projects" program; in fact, DSN expects to save money in the future by using the experiment results.

INTERWORKS is tightly integrated into the present Company effort which pursues business improvement. With this scope, some general projects/processes have been activated across AMS, such as:

- BEM, which is a program of the European Foundation for Quality Management (EFQM) aimed at achieving and maintaining excellence;
- CMM, which is an internationally recognised method of assessing how the software production is managed and supporting its improvement;
- Lifecycle Management, which is for setting best practices;

In addition to these, very general methods, specific technologies/methodologies/techniques were used during the experiment. They are described within the paper (for example, see nr. 3a)

2. Project Overview

INTERWORKS was structured in two main phases subdivided in different lines.

Phase 1

- Training: to change the company culture and prepare new skills (see nr. 4)
- Analysis of critical aspects: to understand the issues and face new challenges (see nr. 3c)
- Process modelling: to understand the present process and design the new one (see nr. 3b)
- Tools evaluation & selection: to standardise across the company and choose the most appropriate tools (see 3d)
- Cooperative environment: to set up a unique INTRANET portal to access the resources and interact (see nr. 3e) named COIN (Concurrent Operations INterface).

Phase 2

- Experiment execution: to apply INTERWORKS by using a significant case (see nr. 4a)
- Metrics setting: to translate our understanding in quantitative appropriate values (see nr. 4b)

The process feedback is constituted by the results evaluation (see nr. 6) and corrective actions identification (see nr. 7). It means that the improvement process works cyclically on a trial-correct basis. The effectiveness of the feedback depends on the metrics appropriateness. This is a crucial point with respect to the objectives and the project's capability of achieving them.

The project impacted on:

- culture
- organization
- technical matters
- skills
- business

The evaluation of these aspects is in nr. 8a.

3. Reengineering

a) Methodology

In order to reform SW design and development process, a specific process based on the re-engineering methodology named PROBE® was chosen because of its wide and successful application within the U.S. department of defence.
This methodology is based on team-building, process modelling and performance measurement techniques, which enable and facilitate the identification and the implementation of organisational and operational changes.
A team of professionals, with different skills, strongly sponsored by the top management and supported by PROBE expert consultants -- with the role of trainer, facilitator and information catalysts -- modelled the Reference and Target software design and development process.
These process models were used during the project to identify areas for improvement, to evaluate problems, to define new solutions, to plan the measurement baseline, to teach the new working practices and to monitor their application.

b) Process

Process definition and modelling are essentially activities of analysis and documentation which make it possible to increase knowledge of the process and to determine the Company behaviour by gathering all relative information (activities, organisation, business regulations, time sequences, logical relationship, data, and so on) in a patterned way.

According to PROBE® approach, the analysis and the modelling of the processes were carried out using IDEF 0, IDEF 3 and IDEF 1X.

The integrated use of these three techniques made it possible to represent in a complete way the process describing:

- the activities and their relationship in terms of time (synchronism, parallelism, etc.) and logic connections;
- the activities and their relationship in terms of input/output, controls which determine the activities models and mechanisms showing the means by which activities are performed;
- data and their relationships, which contribute to constitute the Company knowledge and business regulations.

These techniques made it possible to set a common vocabulary and, as a result, the models become a live reality, which enhanced human-to-human communication, besides providing the organisational and I.T. re-engineering specifications.

c) Critical aspects – Proposed changes

The main critical aspects identified along the process analysis can be synthesised as follows:

- obsolescence and ineffectiveness of "waterfall" approach for software design and development: the unsteadiness and the continual variability of software requirements are characteristics proper to our domestic market and they cannot be eliminated enforcing the design reviews. Moreover, the requirements stabilisation process finished only at the end of the design and development phases. Serialise and delay the different phases, waiting for customer requirements freezing, results in a wasting of time;
- loss of strategic competencies; the down-sizing of engineering departments and the uncontrolled outsourcing of software design and development activities, have progressively weakened the Company competencies;
- lack of concurrent engineering practices and tools;
- inadequacy in software design and development: the actual process is not suffuciently based on the use of techniques for product architecture management, configuration management and software reuse.
- difficulties in making compatible the electronic data exchange typical of concurrent operations and the security issues as stated by the National Authority for Security.

In order to overcome some of the critical aspects and to improve the effectiveness without penalising the customer needs in terms of product customisation, the PIE team defined and implemented organisational and methodological changes to introduce:

- evolutionary and iterative software engineering approach;
- "object-oriented" techniques and tools;
- software product planning;
- software requirements tracking;
- reusable software component planning, development and maintenance;
- software configuration management;
- protocols, rules and tools for knowledge sharing and co-operative work.

Figure FMAN.1 shows the "as is" process, i. e. how the process "CSCI Software Requirements Analysis" usually works inside AMS/DSN. Figure FMAN.2 shows the "to be" process, i. e. the process objective of the experiment which takes into account the described changes.

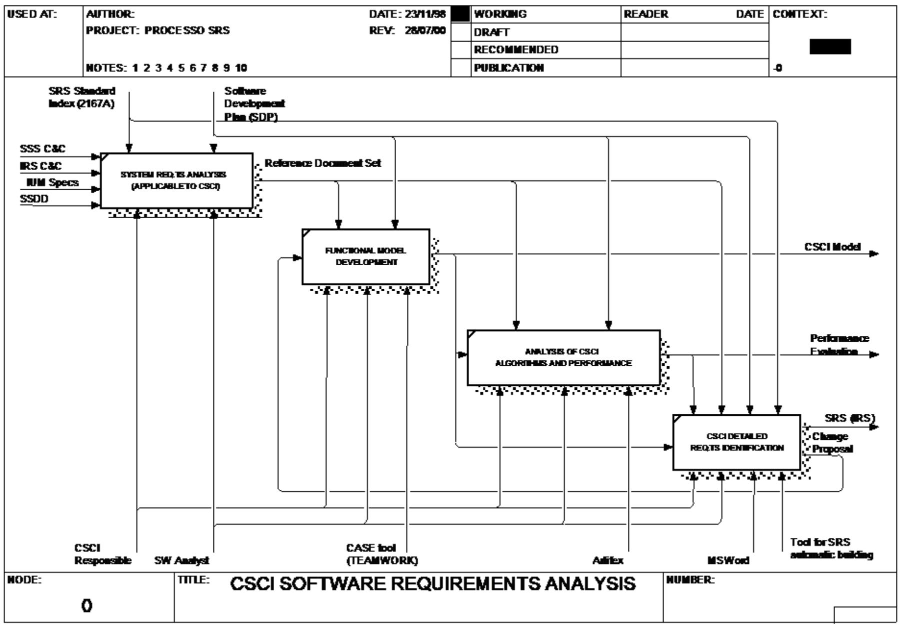

USED AT:
AUTHOR:
PROJECT: PROCESSO SRS
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 23/11/98
REV: 28/07/00
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
-0
SRS Standard Index (2167A)
Software Development Plan (SDP)
SSS C&C
IRS C&C
HJM Specs
SSDD
SYSTEM REQ.TS ANALYSIS (APPLICABLE TO CSCI)
Reference Document Set
FUNCTIONAL MODEL DEVELOPMENT
CSCI Model
ANALYSIS OF CSCI ALGORITHMS AND PERFORMANCE
Performance Evaluation
CSCI DETAILED REQ.TS IDENTIFICATION
SRS (IRS)
Change Proposal
CSCI Responsible
SW Analyst
CASE tool (TEAMWORK)
Adifex
MSWord
Tool for SRS automatic building
NODE:
0
TITLE:
CSCI SOFTWARE REQUIREMENTS ANALYSIS
NUMBER:

Fig. FMAN.1

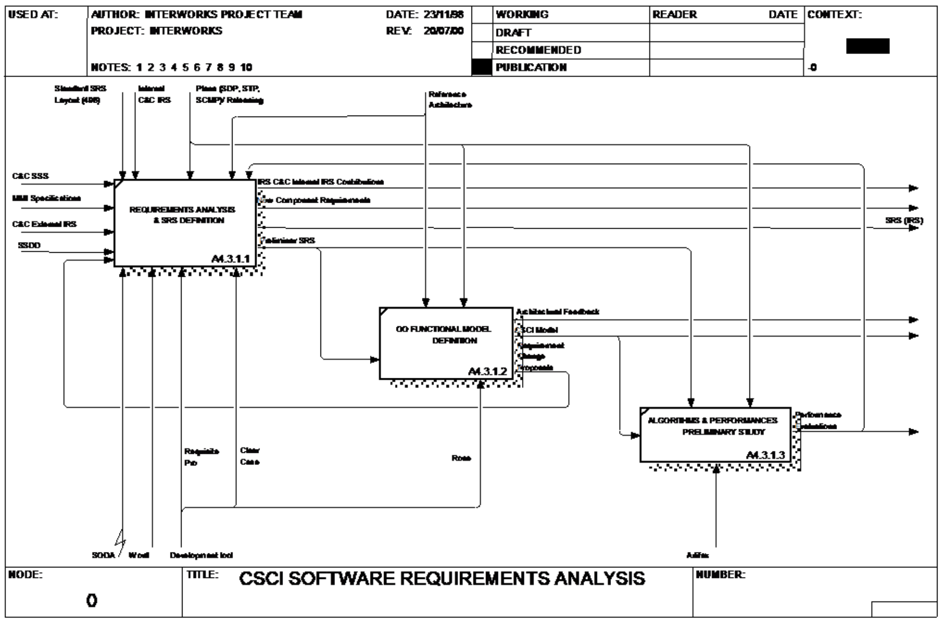


Fig. FMAN.2

d) Tools

After a deep research and analysis performed during the first phase in order to evaluate the different opportunities, the set of selected tools was stated as follows:

- MS-Word ©
- Net-meeting ©
- Rational Rose ©
- Requisite Pro ©
- Clear Case ©
- Soda ©

e) Cooperative environment - COIN

COIN is an application designed to support the needs of concurrent work and teams. The general architecture for COIN was defined in strict analogy to the new concepts introduced by INTERNET to support workgroup in cyberspace. It is such a Portal which allows access to resources, to working areas, to folders. It works by sharing data, communicating, co-ordinating and ensuring security and privacy.
The functional architecture of COIN is based on the elements specified below.

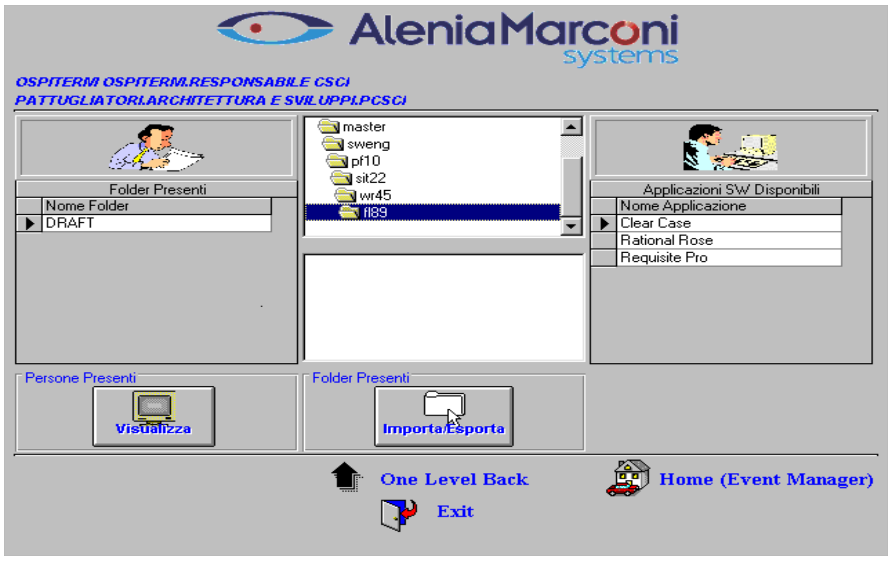


Fig. FMAN.3

• **WORKING ROOM**: virtual sites where several people can meet and work with data, documents, folders, software applications available on sites
• **LAYOUT**: set of working rooms, which make it possible to carry out a specific process
• **SITES**: virtual sites where layouts are made operative
• **FOLDER**: group of files identified by a single name valid for the whole network and, therefore, independent from working rooms;
• **PLATFORM**: HW devices and SW applications set supporting sites, working rooms and basic services.

With regard to SW applications and technological aspects, the leading patterns followed can be summarized as follows:
• use of COTS SW applications
• use of CALS STANDARDS
• single integrated company network linking the Company's offices in Rome, Genoa and Taranto.
• browser equipped client to access services and SW applications
• integrated set of specialized servers:
  - WEB SERVER
  - SQL SERVER
  - DATA WAREHUOSE SERVER
  - PDM SERVER
  - "SAP" SERVER
  - APPLICATION SERVER
Fig. FMAN.3 shows an example of COIN.

4. Training

Particular attention was paid to professional training. This is because INTERWORKS's objective was to impact on company culture and personnel skills. Therefore, two sets of courses were planned: the first dedicated to general approach, critical topics and methodology, the second to the subjects more strictly of interest for the INTERWORKS people.
The following courses were held:

1.1.1.1	**Main subjects**	**Person-Days**
PROBE	The Business Process Reengineering Approach IDEF0 IDEF3 IDEF1X (all the steps with practical stages)	2 sessions, each one 4 days long. Participants: 14 senior technicians
Computer Support Cooperati ve Work	Cooperative work Virtual Corporation (security and quality) Co-authoring Audio-video-conference e-mail Configuration Management Distributed design WEB	2 sessions, each one 1 day long. Participants: 14 senior technicians
Virtual Team Working	Virtual Team Co-operative Work	2 sessions, each one 2 days long. Participants: 14 senior technicians

There were, also, some presentations of Concurrent Engineering as a Company choice to achieve and maintain excellence.
In addition, practical stages were organised based on three main issues:

- Object Oriented Methodology for software developing
- Network Technologies: INTERWORKS Project Overview
- Suite Rational Usage

The practical stages involved about 30 people for 5 days. The stages had a specific section dedicated to the COIN environment and its usage with practical sessions. A short "user manual" was published in the DSN WEB site at the section dedicated to the Business Improvement Projects.
Two internal Workshops, one at Rome and one at Genoa, were held for a wide audience in order to present and discuss the Concurrent Operations approach which includes virtual teams dedicated to software production.
The training required about 100 days to be prepared; the manuals were published on the INTRANET WEB site.

5. Experiment execution and evaluation

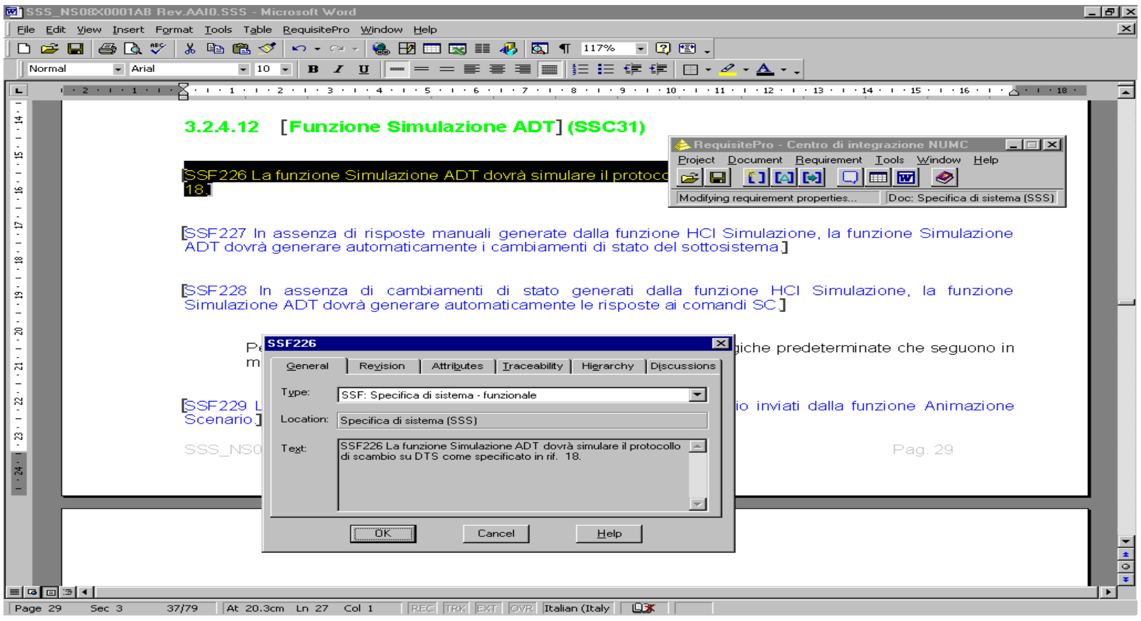
a) Baseline Project

Within AMS/DSN, the Software development process is structured in several phases. All of them are fully documented and reviewed; these phases include Software Requirements Analysis, Preliminary Design, Detailed Design, Coding, Integration, Host/Target Testing and delivery of all required reports/documents.
Different teams are allocated to different projects; each team is lead by a Technical Leader. If the software component is part of a wider system, the Technical Leader reports to a System Leader, who is responsible for the hardware and software integrated supply.
The economic and commercial aspects are under the overall responsibility of a Program Manager.
Software Quality Assurance (SQA) and Software Quality Control (SQC) are staff functions; they allocate personnel to Software projects and independently report to general management.
Software testing is performed by the developers, who have the responsibility of preparing the Test Plan and the Test Procedures; the execution of testing is carried out under SQC witnessing.
Because AMS/DSN is an official supplier of the Italian M.o.D. (Ministry of Defence), its Software development process has been qualified following the NATO AQAP-13 Standard. World wide known standards are largely adopted, such as DOD-STD-2167A and MIL-STD-498.
When a Software development is externally subcontracted, procedures exist to verify and ensure the correctness of the process.
The activity progress is assessed on a periodic basis. If needed, corrective actions are proposed and, once agreed upon by all parties, the project plan is updated. If the corrective actions impact on any portion of already delivered products, the change is formally processed by the appropriate board (Change Control Board) and the Configuration Control procedure activated.
The personnel operate in three factories, located in Rome, Genoa and Taranto. Moreover, in case of work peaks, part of the software development is assigned to different subcontractors, who carry out the development process at their plants.
At the end of the development, the final hardware and Software integration process is normally performed on site, in particular on naval units at national or foreign shipyards.
AMS/DSN chosen a Software development project belonging to the domain of Command & Control systems as the Baseline Project for INTERWORKS.
The project concerns design and development of a complete Command and Control System for naval application, with a strong percentage of functionality implemented by software applications. The Baseline Project applies to a market area in which AMS/DSN is putting the major effort into conquering new business opportunities.
The project is a typical example of geographically distributed development and it is planned to be carried out using ADA language and MSTD-498 standard.
The project cost breakdown structure indicates that the main costs are allocated in the cost category "Software development". The considered lifecycle includes system management, product planning, product research, engineering design, documentation, software production, testing, hardware procurement, system integration, start-up and validation.

b) Application

INTERWORKS was applied, from July 1999 to March 2000, at the Simulation Centre, which is part of the Baseline Project.
The process affected by the experiment is a step of the software development: the Software Requirements Specifications issue for each CSCI, as resulted from the subsystem breakdown. It means that all the activities which lead to the issue of the SRS were performed by using the new working practices, the selected tools and the implemented COIN.
In particular, the CSCI involved are 17, six of the Real World Simulation and Testing Facilities and 11 of the Sub-system Simulators and Stimulators.
The project team is composed of 19 AMS-DSN and SSI (subcontractor) professionals that are working in the factory sites located in Genoa, Rome and Taranto.
Each one of the professionals is working in strict cooperation with the others, sharing the documents in progress and using the "COIN" application and the Rationale® Suite functionalities.

By using Requisite Pro® the requirements are set on a data base and it is possible to trace them and easily obtain cross correlation matrixes. Fug. FMAN.4 shows an example of a functional requirement at system level. Such a management is compliant with the CMM recommendations at level 2. In addition, it is possible to obtain a view of attributes for each requirement (Fig. FMAN.5) or the relationship between different requirements (Fig. FMAN.6).



SSS_NS08X0001AB Rev.AAI0.SSS - Microsoft Word

3.2.4.12 [Funzione Simulazione ADT] (SSC31)

[SSF226 La funzione Simulazione ADT dovrà simulare il protoco
18.]

[SSF227 In assenza di risposte manuali generate dalla funzione HCI Simulazione, la funzione Simulazione ADT dovrà generare automaticamente i cambiamenti di stato del sottosistema.]

[SSF228 In assenza di cambiamenti di stato generati dalla funzione HCI Simulazione, la funzione Simulazione ADT dovrà generare automaticamente le risposte ai comandi SC.]

Pe giche predeterminate che seguono in
m

[SSF229 L io inviati dalla funzione Animazione
Scenario.]

SSS_NS0 Pag. 29

RequisitePro - Centro di integrazione NUMC

Modifying requirement properties... | Doc: Specifica di sistema (SSS)

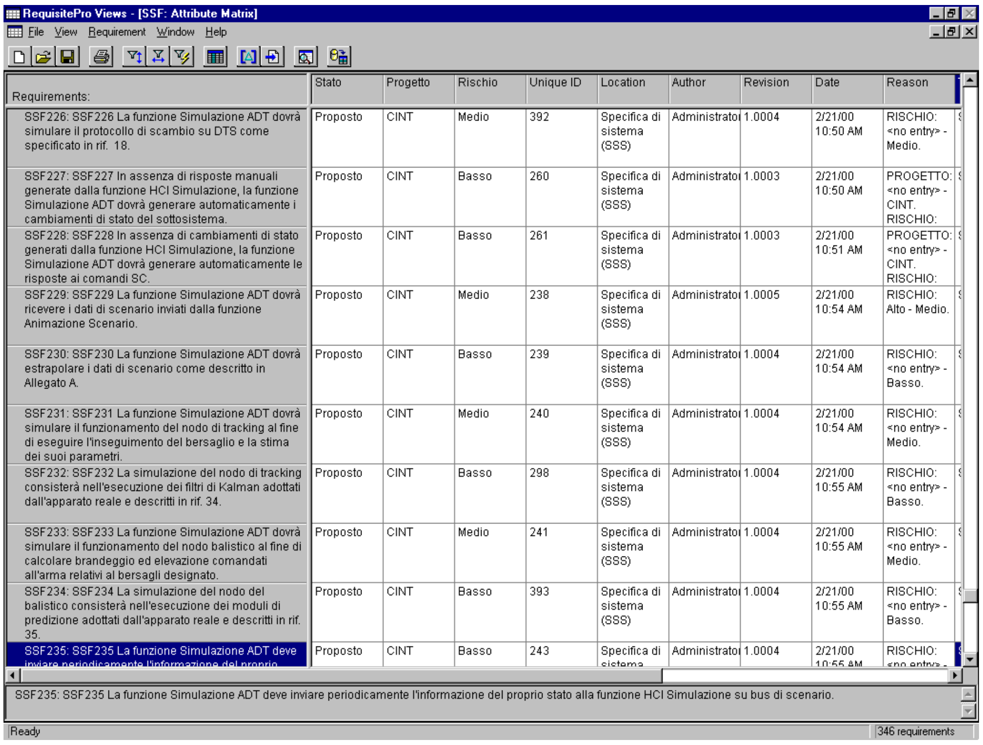
SSF226

Type:	SSF: Specifica di sistema - funzionale
Location:	Specifica di sistema (SSS)
Text:	SSF226 La funzione Simulazione ADT dovrà simulare il protocollo di scambio su DTS come specificato in rif. 18.

Page 29 Sec 3 37/79 At 20.3cm Ln 27 Col 1 REC TRK EXT OVR Italian (Italy

File Edit View Insert Format Tools Table RequisitePro Window Help
117%
Normal Arial 10
Project Document Requirement Tools Window Help
General Revision Attributes Traceability Hierarchy Discussions
OK Cancel Help

Fig FMAN.4



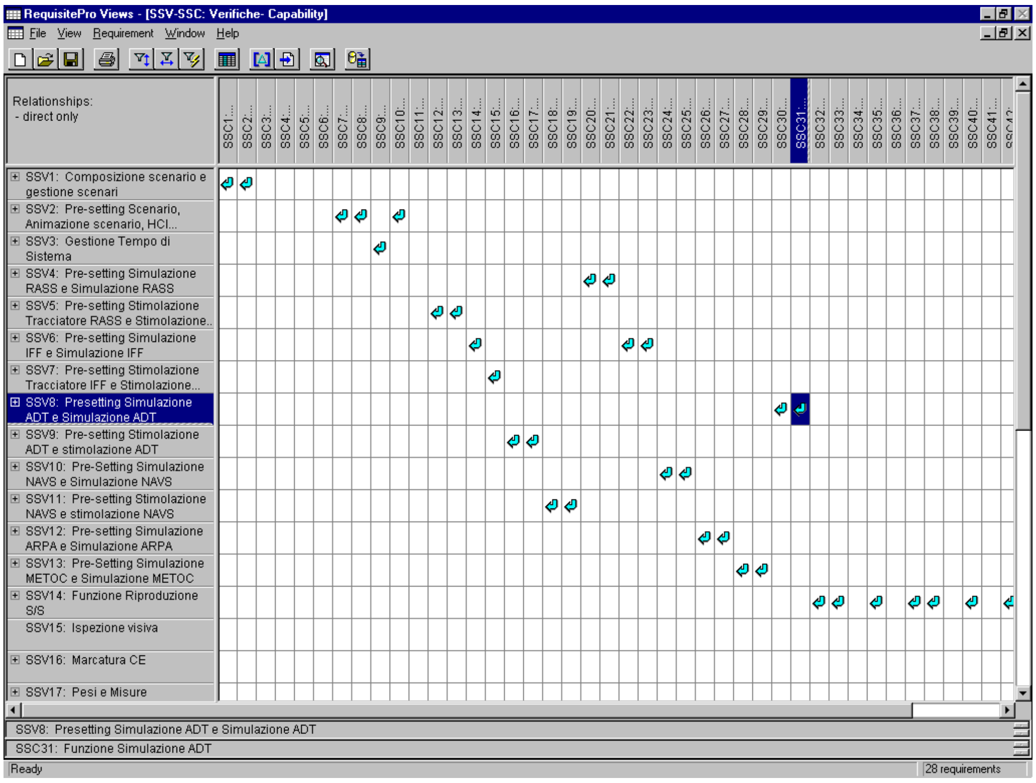
RequisitePro Views - [SSF: Attribute Matrix]

Requirements:	Stato	Progetto	Rischio	Unique ID	Location	Author	Revision	Date	Reason
SSF226: SSF226 La funzione Simulazione ADT dovrà simulare il protocollo di scambio su DTS come specificato in rif. 18.	Proposto	CINT	Medio	392	Specifica di sistema (SSS)	Administrator	1.0004	2/21/00 10:50 AM	RISCHIO: <no entry> - Medio.
SSF227: SSF227 In assenza di risposte manuali generate dalla funzione HCI Simulazione, la funzione Simulazione ADT dovrà generare automaticamente i cambiamenti di stato del sottosistema.	Proposto	CINT	Basso	260	Specifica di sistema (SSS)	Administrator	1.0003	2/21/00 10:50 AM	PROGETTO: <no entry> - CINT. RISCHIO:
SSF228: SSF228 In assenza di cambiamenti di stato generati dalla funzione HCI Simulazione, la funzione Simulazione ADT dovrà generare automaticamente le risposte ai comandi SC.	Proposto	CINT	Basso	261	Specifica di sistema (SSS)	Administrator	1.0003	2/21/00 10:51 AM	PROGETTO: <no entry> - CINT. RISCHIO:
SSF229: SSF229 La funzione Simulazione ADT dovrà ricevere i dati di scenario inviati dalla funzione Animazione Scenario.	Proposto	CINT	Medio	238	Specifica di sistema (SSS)	Administrator	1.0005	2/21/00 10:54 AM	RISCHIO: Alto - Medio.
SSF230: SSF230 La funzione Simulazione ADT dovrà estrapolare i dati di scenario come descritto in Allegato A.	Proposto	CINT	Basso	239	Specifica di sistema (SSS)	Administrator	1.0004	2/21/00 10:54 AM	RISCHIO: <no entry> - Basso.
SSF231: SSF231 La funzione Simulazione ADT dovrà simulare il funzionamento del nodo di tracking al fine di eseguire l'inseguimento del bersaglio e la stima dei suoi parametri.	Proposto	CINT	Medio	240	Specifica di sistema (SSS)	Administrator	1.0004	2/21/00 10:54 AM	RISCHIO: <no entry> - Medio.
SSF232: SSF232 La simulazione del nodo di tracking consisterà nell'esecuzione dei filtri di Kalman adottati dall'apparato reale e descritti in rif. 34.	Proposto	CINT	Basso	298	Specifica di sistema (SSS)	Administrator	1.0004	2/21/00 10:55 AM	RISCHIO: <no entry> - Basso.
SSF233: SSF233 La funzione Simulazione ADT dovrà simulare il funzionamento del nodo balistico al fine di calcolare brandeggio ed elevazione comandati all'arma relativi al bersagli designato.	Proposto	CINT	Medio	241	Specifica di sistema (SSS)	Administrator	1.0004	2/21/00 10:55 AM	RISCHIO: <no entry> - Medio.
SSF234: SSF234 La simulazione del nodo del balistico consisterà nell'esecuzione dei moduli di predizione adottati dall'apparato reale e descritti in rif. 35.	Proposto	CINT	Basso	393	Specifica di sistema (SSS)	Administrator	1.0004	2/21/00 10:55 AM	RISCHIO: <no entry> - Basso.
SSF235: SSF235 La funzione Simulazione ADT deve inviare periodicamente l'informazione del proprio	Proposto	CINT	Basso	243	Specifica di sistema	Administrator	1.0004	2/21/00 10:55 AM	RISCHIO: <no entry> -

SSF235: SSF235 La funzione Simulazione ADT deve inviare periodicamente l'informazione del proprio stato alla funzione HCI Simulazione su bus di scenario.

Ready | 346 requirements

Fig. FMAN.5



RequisitePro Views - [SSV-SSC: Verifiche- Capability]

Relationships:
- direct only

SSC1... SSC2... SSC3... SSC4... SSC5... SSC6... SSC7... SSC8... SSC9... SSC10... SSC11... SSC12... SSC13... SSC14... SSC15... SSC16... SSC17... SSC18... SSC19... SSC20... SSC21... SSC22... SSC23... SSC24... SSC25... SSC26... SSC27... SSC28... SSC29... SSC30... SSC31... SSC32... SSC33... SSC34... SSC35... SSC36... SSC37... SSC38... SSC39... SSC40... SSC41...

SSV1: Composizione scenario e gestione scenari
SSV2: Pre-setting Scenario, Animazione scenario, HCI...
SSV3: Gestione Tempo di Sistema
SSV4: Pre-setting Simulazione RASS e Simulazione RASS
SSV5: Pre-setting Stimolazione Tracciatore RASS e Stimolazione...
SSV6: Pre-setting Simulazione IFF e Simulazione IFF
SSV7: Pre-setting Stimolazione Tracciatore IFF e Stimolazione...
SSV8: Presetting Simulazione ADT e Simulazione ADT
SSV9: Pre-setting Stimolazione ADT e stimolazione ADT
SSV10: Pre-Setting Simulazione NAVS e Simulazione NAVS
SSV11: Pre-setting Stimolazione NAVS e stimolazione NAVS
SSV12: Pre-setting Simulazione ARPA e Simulazione ARPA
SSV13: Pre-Setting Simulazione METOC e Simulazione METOC
SSV14: Funzione Riproduzione S/S
SSV15: Ispezione visiva
SSV16: Marcatura CE
SSV17: Pesi e Misure

SSV8: Presetting Simulazione ADT e Simulazione ADT
SSC31: Funzione Simulazione ADT
Ready
28 requirements

Fig. FMAN.6

c) Metrics

In order to evaluate the experiment results, the PIE team defined a specific measurement process driven by goals and issues related to three different kinds of management area:
1. Program/Project Management
2. Process Management
3. Product Management

- Program/Project Management

The objectives of program/project management are to set and, then, meet achievable commitments regarding costs, schedule, quality and product performances delivered within business projects.
Program/Project Management function represents the customers and the shareholders so, as a consequence, it is the first client to satisfy when introducing new working practices.
For this reason, the function was involved in the measurement process.
In particular, the experiment was planned and controlled using project management techniques (WBS, GANTT, CISCSC) supported by the ERP in order to quantify the saving in terms of effort and costs.

- Process Management

The objectives of process management are to ensure that the processes within the organisation are performing as expected.
The target process has been evaluated respect to the following aspects:
- Performance
- Stability
- Compliance

The **performance** measurement is focused on how the target process works compared to the past. Also for this reason the PIE team identified the reference model characterised by the parameters defined through the analysis of the programs developed in the past.
In particular the performance measurement is focused on the amount of effort and costs spent to produce the SRS document compared to the number of requirements.
Stability is the capability of a process to be predictable; we have a stable process when all assignable causes of variations have been removed and prevented.

In order to meet an acceptable level of stability a "red team" -- composed of quality assurance and software engineering senior representatives -- controlled the execution of the experiment to identify the problems eventually produced for assignable causes (inadequately trained people, tool failures, altered methods and so forth) and to define, according to the PIE teams, the suitable corrective actions.
Compliance is achieved when standards and practices exist and are followed. Compliance also means that the process is supported adequately and that the organisation is capable of executing the process.
The aspects of compliance under evaluation are:
- adherence of the different CSCI processes to the target process
- suitability and use of tools and procedures.

At the end of the assigned task, people involved in the experiment must fulfil a form in order to point out the differences between the target process and the performed process.
- Product Management
The product management discipline is applied to four strictly correlated processes:
- Product Planning
- Meta-Architecture Management
- Product Configuration Management
- Software Re-use

These processes support and lead software design and development in order to maximise component re-use and product modularity.
This task is entrusted to the software configuration manager who must control:
- the number of Software Component Unit shared among the CSCI
- the number of engineering change requests and orders for each CSCI/CSU.

6. Results

a) Program/Project Management

The results in terms of savings are positive.
The following table shows the planned data compared to the final effort and costs spent to release the SRS documents for the different CSCI.

	PLANNED	ACTUAL	SAVINGS	
TOTAL EFFORT (h)	3.100	2.750	350	11%
TOTAL LABOUR COSTS (ECU)	104.000	92.500	11.500	11%
TOTAL TRAVEL COSTS (ECU)	15.600	7.600	8.000	51%
TOTAL COSTS (ECU)	119.600	100.100	19.500	16%

b) Process Management
- Performance

The comparison between the reference process parameters and the results of the experiment are highlighted in the following table

	h/Req.	% Travel Costs
REFERENCE PROCESS	4	27%
EXPERIMENT	3.4	8%
SAVINGS	15%	70%

- Stability

After the experiment and the corrective actions, the SRS process looks stable and well defined.
The programs started after the end of the experiment are using, if appropriate, the new working practices. The dissemination process has also been activated into the other AMS Divisions.
- Compliance
The level of compliance reached by the different SRS process as performed during and after the experiment is quite considerable.
All the professionals who are using the new procedures and the new tools foreseen in the target process, have found the solutions to the usual problems of their work.
In particular the key aspects that satisfied their expectations were:
- the new functions of communication and data/document sharing
- the suitability of the tools
- the performances of the network.

At this moment some difficulties and unresolved problems are related to the application of the support process such as:
- product planning
- meta-architecture management.

These processes involved some other skills and functions out of the software factory and the different roles must still be set-up.

c) Product Management Area

The results in terms of software reuse will be measured only after some similar projects will be performed.

7. Corrective Actions

The main sources of corrective actions were two: the red team -- charged to overview the process stability -- and the technical meetings, held during the project development. The following corrective actions have been implemented until now:

- the Requisite Pro database has been organised in order to manage separately the system requirements and the requirements of each CSCI;
- the metric measurements were performed by SoDA;
- the C++ code was connected to the UML models defined within the Rational Rose application in order to increase the adherence between the code e the OO model, eliminate the redundancy in code lines, homogenise the structures of the different CSCI, facilitate the design and automate the modification of the documents;
- the use of tables within the MS Word documents was limited because of the difficulty in handling them by Requisite Pro;
- the use of Requisite Pro was extended to the System Requirements analysis and specification in order to save time during SRS having all the input documents developed by Requisite Pro;
- an intranet newsgroup open to the whole Company in order to share the experience was started.

8. Final considerations

a) Impacts

INTERWORKS is a re-engineering project aimed at changing the way DSN is used to develop software. It took about two years, impacting on various aspects as reported below.
<u>Culture</u>. - It was clear, at the beginning of the project, on the basis of the previous experience, that it is useless change tools if the methods/behaviour are not changed. The tools can help, but they do not

represent the most important factor in order to establish a new process. It is more important and more challenging to overcame the cultural inertia.
Two aspects found particular resistance: (a) implement a change if requires the continuous and active collaboration of people; (b) keep doing workgroup within a virtual environment.
The right solution was a well-structured training program strongly sponsored by the top management. PROBE and VIRTUAL TEAM courses helped in reaching new skills and activating new practices.
<u>Organisation.</u> - Three processes required by INTERWORKS had a significant impact on organisation: (1) product planning, (2) software configuration management and (3) reusable software components planning, development and maintenance. These processes impacted the subcontractor's organization too, starting a change, which should increase our capability in satisfying requirements.
<u>Techniques.</u> – The target process required new techniques in software requirements area; it was particularly due to the adoption of object oriented approach and concurrent operations. Again, training and sponsorship were the correct solutions.
<u>Skills.</u> – The target process required new skills impacting on traditional roles such as software architect, software analyst and software developer. The change was afforded by redesigning the roles according to the OO approach and widening the competences in software management and workgroup techniques area.
<u>Business.</u> – It is appropriate, on the basis of the reported results, to foreseen an improvement in Company's competitiveness. Nevertheless, at the moment, it is really too early to evaluate the impact in terms of prices and market share. It is necessary to apply the new process to the overall software life cycle and to different projects.

b) Lessons learned

INTERWORKS experience was rich of suggestions for future application of innovative projects. We intend to highlight three lessons:

- human to human communication is the first problem to afford and solve. Workgroup is based on structured communications among individuals. As matter of fact, to obtain "good" communication, the group needs to share not only the tools but, which is more important, the objectives and the procedures.
- as many chemical process, the reengineering one needs a specific amount of power to be activated. The no return point must bee reached, so the appropriate amount of resources (people and money) should bee planned and delivered at the right time.
- the continuity of the innovative action is important. It means success confidence: the people involved are convinced and the plan works well (all is under control).

c) Conclusions

We can conclude that the economical and methodological results of INTERWORKS represent a significant initial step in improving AMS/DSN competitiveness.
Extrapolating the results on the annual amount of software developed by the division, it is ease to reach big numbers in matter of savings. In addition, we expect a significant reduction of the "time to market".
In order to consolidate the experiment results, we are sponsoring a large program to disseminate the methodology, apply the process, and measure the results.
The INTERWORKS group is honoured to thank the ESSI-PIE (European Systems and Software Initiative-Process Improvement Experiment) for funding, help and suggestions. The author thanks CESVIT and TOPS (Towards Organized Software Processes) Program for effective help and useful suggestions. Finally, we intend to mention the partners in developing the project: SIA SpA (Torino-Italy) and Thema Associated (Milano-Italy).

References

[1] Ansoff,H.Iz. “The Firm of the future”, Harvard Business Review, Vol. 43 (5), pp. 162-78, (1965)
[2] Appleton,D. “PROBE, Principles and Rules of Business Engineering” (1995)
[3] Baroni,F. “Concurrent Operations:Project Methodology”, CALS Expo INTERNATIONAL- Tokyo, (1997)
[4] Berard “Essays on Object Oriented Engineering”, 1993
[5] Booch, “Object solutions managing for Object-Oriented projects”, 1996
[6] Child,J. “Management and organisational factors associated with company performance”, Journal of Management Studies, Vol. 11 (October), pp. 175-89, (1974)
[7] Davemport,T.E. & Short, J.E. “The new industrial engineering: information technology and business process redesign” Sloan Management Review, Vol. 31 No. 4 1990
[8] Hammer,M. “Re-engineering work: don’t automate, obliterate”, Harvard Business Review, Jul-Aug, (1990)
[9] Hammer,M. & Champy, J. “Re-engineering the corporation, Harp Business”, New York 1993
[10] Kay,J. Foundations of Corporate Success: How business strategies add value” Oxford University Press, (1993)
[11] Lorenz, “Object Oriented Software Development”, 1993
[12] McGregor, “Sykes - Object Oriented Software Development Engineering for Reuse”, 1994
[13] Maull, R.S. Childe, S.J. Bennett,J. Weaver,A.M. Smart,P.A. “Different types of manufacturing process and IDEF0 models describing standard business process” Working Paper WP/GR/J95010-6, University of Plymouth, March 1995
[14] UNICOM Conference “Improving Business Performance Through Effective use of IT”, London 7-9 June 1994
[15] Jacobson, “Object Oriented software engineering”, 1992
[16] Jacobson, Griss – “Software Reuse”, 1997

Felicio MANZO, dr,
Alenia Marconi Systems S.p.A.
via Tiburtina 1231, 00131 ROME-ITALY
tel. +39-10-6546-240;
e-mail: fmanzo@aleniasystems.finmeccanica.it

Mr. F. Manzo is manager, since ’99, of the “Process and Information Technology” Unit of Alenia Marconi Systems – Naval Systems ITA Division. The Unit has the objective to improve Company’s business by re-engineering the divisional processes. The improvement could include, but is not limited to, the usage of information technology systems.
Previously, Mr Manzo, since ‘76, has worked for other companies in iron and steel, nuclear, engineering and advanced software development industrial areas. He held positions for computer systems design & development and Program Management in Italy and USA. He managed the Research Consortium “SESPIM” for advanced software development dedicated to the Defence Industries. Actually, Mr Manzo serves the SESPIM Consortium Board.

ALENIA MARCONI SYSTEMS

Alenia Marconi Systems is an equal shares joint venture between BAE Systems of the UK and Finmeccanica of Italy. The Company is a major force in European defence and electronics with a turnover of over ECU 1.9 Bn and an established international customer base in over 100 countries.
In particular AMS is a leading ground and naval radar supplier in the world, a leading supplier of missile systems in Europe, an acknowledged expert in land and naval command and control systems, a leading supplier of simulation systems and a worldwide provider of customer support and training systems.

AMS comprises the following divisions: Land Systems, Air and Traffic Management Systems, Missile Systems, Operations and Customer Support.

Role and Impact of Feedback and System Dynamics in Software Evolution Processes and their Improvement

Meir M. Lehman
Goel Kahen
Juan F. Ramil
Imperial College, London, UK

Abstract

The FEAST/1 and FEAST/2 projects followed formulation of a hypothesis that apart, perhaps, from primitive processes, software processes are complex multi-loop, multi-agent, multi-level feedback systems. In order to achieve sustained process improvement processes must, therefore, be treated as feedback systems. The results of the FEAST studies (see http://www-dse.doc.ic.ac.uk/~mml/feast) include a set of models that reflect the behaviour of evolutionary attributes of real world industrial processes and their products. They also include further support for the laws of software evolution and generalisation of an approach to the development of quantitative black-box and white-box (system dynamic) models of evolution processes. Taken together they reinforce the conclusions of studies reached during the 70s and 80s. This paper summarises some of the concepts developed and lessons learnt to date, emphasising what is relevant in the context of process improvement.

Introduction

Software evolution includes the fixing, enhancement, adaptation and extension of a software system; that is, it encompasses all aspects of software maintenance, as generally understood. Interest in this phenomenon is now widespread. Most of the work addressing the area concentrates, however, on the *means* whereby the ease and reliability of evolution is increased and its cost reduced. This focus on means is generally undertaken without sound understanding of *why* evolution occurs, *why* it is inevitable, *what* are its key attributes and how these could be controlled and exploited to achieve a process that meets the challenges of evolution. Answers to some of the above questions emerged in a series of metrics-based studies of software systems conducted during the late 60s, and the 70s, [leh69,85]. These led, *inter alia*, to the laws of software evolution [leh74,85,fea00b], a concept of *anti-regressive* work [leh74], the *SPE* program classification [leh85] and a Principle of Software Uncertainty [leh89,90].

In 1993, while considering why it is generally so difficult to achieve sustained software process improvement, one of the authors (mml) hypothesised that this may be related to the *feedback*[1] nature of such processes. In so doing, he recalled a study and observation made nearly thirty years ago when, discussing the *ripple* observed in a plot over release sequence numbers (RSN) of the OS/360-370 growth trend [bel72], as in figure MML.0.

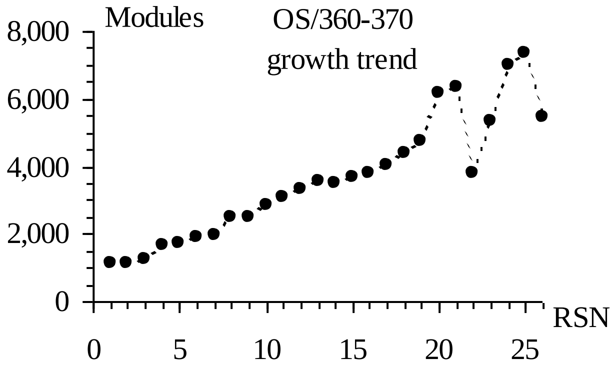


Fig. MML.0: IBM OS/360-370 Growth Trend

The ripple superimposed on an initial underlying linear[2] growth trend was then interpreted as being the result of a self-stabilising process, the consequence of counter-acting positive and negative feedback forces. On the one hand, factors such as the need to overcome limitations encountered when running the system, ambition to achieve greater market share and so on, lead to pressures to *increase* the functional power of the system and, hence, to system growth. On the other hand, constraints emerge that tend to *limit* the achievement of further growth. These include, for example, those imposed by increasing system complexity, neglect of complexity control (i.e., *anti regressive* [leh74] work) and resource availability. The former exemplify positive loops that reinforce a particular behaviour and so lead, in general, to growth or amplification. The latter exemplify negative feedback loops that contribute towards behavioural balance or stabilisation. This analysis led directly to the feedback interpretation referred to above. It was supported by the realisation that the instability observed after RSN 20 was related to excessive positive feedback as suggested by the well above average incremental growth between releases 19 and 20.

Recalling this early observation, he now (1993) proposed the FEAST (*F*eedback, *E*volution and *S*oftware *T*echnology) hypothesis [leh94]. In one of its formulations, this states that, "… with the possible exception of the less *mature*, software processes are multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve sustained process improvement". The theme was discussed in three international workshops at Imperial College, London [fea94,95]. Subsequently, the UK EPSRC funded FEAST/1 (1996-1998) and FEAST/2 (1999-2001) projects have been investigating the hypothesis and its implications. More recently, a fourth international workshop, FEAST 2000, discussed related issues [fea00b].

The projects were set up to achieve a better understanding of the *what* and the *why* of evolution and to determine whether the results obtained during the 70s and 80s were applicable to the processes and technologies of the 90s. The questions being addressed included the following: How may one identify positive and negative feedback loops in software evolution processes and determine the influence of

[1] Feedback is used here in the engineering sense that the output modifies the input.

[2] The FEAST projects have studied the growth patterns of a number of systems and size measures have been fitted to an inverse square model [tur96,fea00a]. This suggests negative feedback dominates.

resulting system dynamics? What is the impact of feedback on such processes and on their products? How may feedback be controlled and exploited to achieve the continuing improvement of such processes and increased likelihood of meeting the challenges of evolution? In searching for plausible answers the FEAST projects have developed a set of models of industrial software processes. This work has been made possible by the provision of empirical data by industrial collaborators and their support in its interpretations. The models represent a means whereby sources of process system dynamics and their strength may be investigated. *Inter alia*, they can also be used to help identify potential process improvements.
This summarises some of the findings and conclusions reached to date with the emphasis on quantitative modelling in the context of process improvement. A list of references covering these and other aspects of the investigation in more details and access to them is provided via links at http://www-dse.doc.ic.ac.uk/~mml/feast.
In connection with what follows it should be noted that the scientific paradigm, as exemplified in figure MML.1, has been followed in FEAST and earlier [leh85] investigations of software evolution. A phenomenon is recognised. Metric data to describe it are obtained. Patterns are identified. Models of that data are built, analysed and validated. The observed phenomenology is interpreted in terms of the models and vice versa and expressed in a formal statement or descriptive text. Understanding of the observed behaviour, the models, and the formal statement or descriptive text are improved in an iterative process of successive refinement.

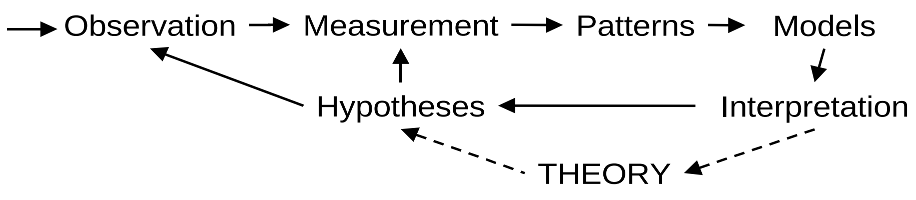


Fig. MML.1: The Investigative Paradigm

Setting the Scene

E-type software is defined as software actively used to solve a problem in a real-world domain [leh85]. The ultimate criterion of satisfaction with an *E*-type system is its acceptability to stakeholders. Whether it is acceptable or not will depend on their needs and desires in relation to the state of the operational domain as the system operates. But *E*-type operational domains are unbounded and dynamic, that is, continually changing. *E*-type specifications are, therefore, necessarily incomplete and will tend to become invalid. Mathematical *correctness* of an *E*-type system with respect to its specification is insufficient to guarantee stakeholder satisfaction.
It follows that continuing evolution is intrinsic to *E*-type software. Most of the software used in business and other organisations to support their operations is of this type. The classification is, therefore most relevant, in particular with respect to the ubiquitous need, since the beginnings of the computer era, for software *maintenance*. Such maintenance is, fundamentally, not due to lack of technical expertise or rigour in the development of software. It is a fact of life. Unlike the maintenance of physical equipment which is mainly targeted at *restoring its original condition*, the purpose of software maintenance is to maintain stakeholder satisfaction, that is to *change the system* to adapt it to changing circumstances, needs and desires. This difference supports the view, long expressed by one of the present authors (mml), that the term *evolution* is more appropriate to describe work undertaken after the first operational release of a software system than is *maintenance*. System evolution is inevitable and must be considered when seeking improvement of the means whereby it is achieved.
How, in practice, is system evolution achieved? As illustrated in figure MML.2, development involves, explicitly or implicitly, bounding of the application domain and definition of an application concept. Sooner or later, stakeholders converge on a common a view of the desired state of the application domain after software deployment, normally reflected in requirement and specification documentation. The latter is a *de facto* abstraction of the desired software in its application domain.

The precise nature or the number of individual process steps that follow is not relevant to the present discussion. The steps shown in MML.2 are one view of the activities necessary to transform an application concept into an operational program. The most important point to emerge from the figure is that, as discussed in the next paragraph, installation of the program in the operational domain, *changes that domain* [leh85]. Installation and operation closes a major feedback loop. Despite differences in technology, methods and process detail between the evolution processes of different organisations all share a common feature, this major feedback loop. The output from the process that is driven by the requirements and specification statements changes the application and the bounds of the domain in which it is to be pursued and which constitute the input to the process.
Once the system is deployed, lessons learned as a result of its being used, in conjunction with exogenous influences such as business, market or legislation changes, will change the application concept and/or the domain bounds. The depicted loop is a major driver of software evolution. There arises, therefore, a continuing need for software adaptation and enhancement. This is generally achieved by a sequence of software releases, versions or update as reflected in the metric and other data that record evolution of the systems [fea00a].

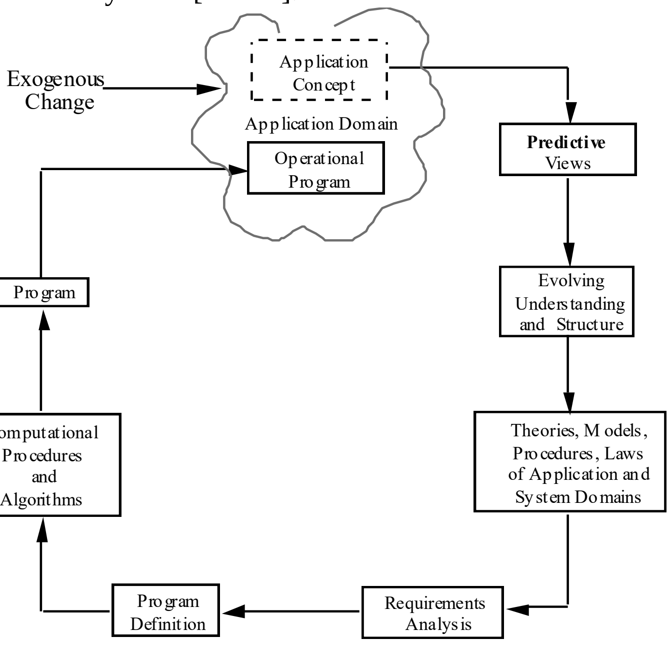

Exogenous Change
Application Concept
Application Domain
Operational Program
Predictive Views
Evolving Understanding and Structure
Theories, Models, Procedures, Laws of Application and System Domains
Requirements Analysis
Program Definition
Computational Procedures and Algorithms
Program

Fig. MML.2 : A Major Process Feedback Loop

The diagram in figure MML.2 is, however, unrealistic in several respects. Evolution processes are, in general, neither strictly sequential nor restricted to technical tasks. An improved representation is provided by MML.3. This seeks to depict the evolution process as a *global* multi-level, multi-loop, multi-agent *process* [leh94].

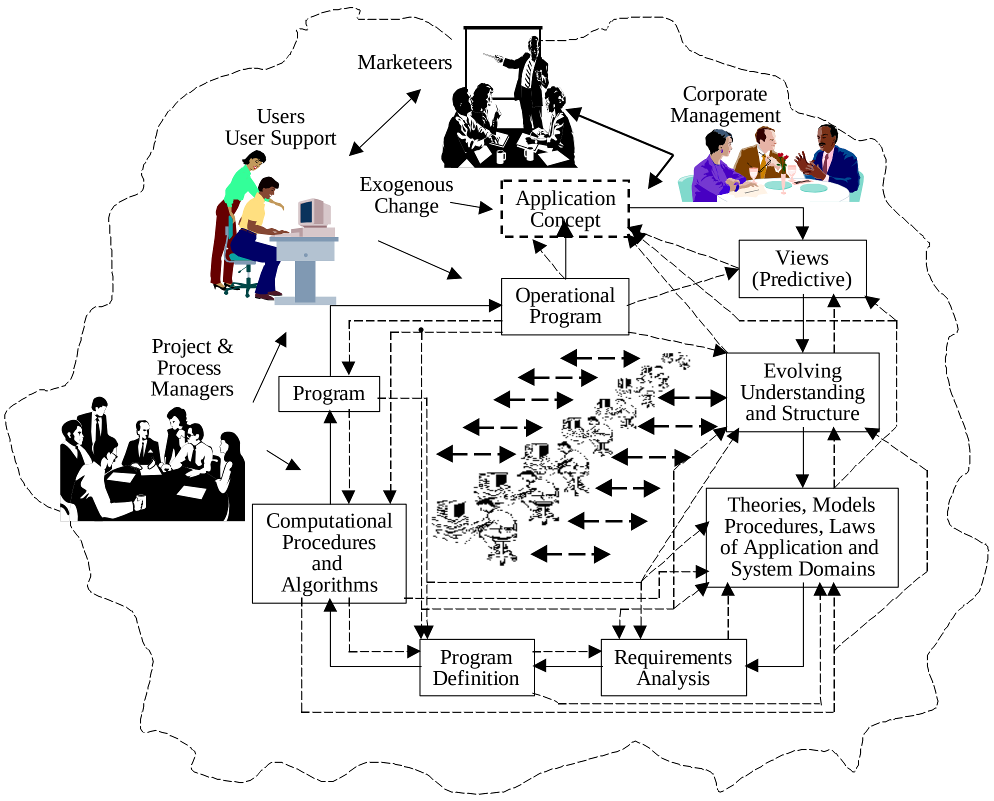


Fig. MML.3 : The Global Software Process

The *global process* illustrated is the totality, the non-linear sum, of all activities required to transform concepts, ideas, definitions, needs specifications, algorithms and models into code and other deliverables through the application of resources. It encompasses the activities of application experts, architects, designers, developers, sales and support personnel, managers, users, etc. Their process related activities are an integral part of the process. All are stakeholders who, in the light of their interests and responsibilities, feed back their perception of needs, expectations, experiences, etc to relevant process agents. Their interactions play a part in driving, directing and constraining the process. All have an impact in and on both process and product.
The fact that the operational system is the product of such global processes imposes numerous challenges when seeking to manage and/or improve either. For example, process changes will have local impact but the global impact may be counterintuitive. Now, with only a few exceptions, e.g., inspections and change management, the emphasis in software process research and practice has focused on the technical process and, in particular, on its forward paths. Feedback mechanisms have, in general, not received much attention despite the fact that they appear to be one of the sources of evolutionary behaviour.
What is known about the evolutionary behaviour of key global process attributes? Investigations over the years have led to the identification of a set of behavioural patterns, termed Laws of Software Evolution, as listed, in their most recent formulation, in Table MML.1 below.

No.	Brief Name	Law
I 1974	Continuing Change	*E*-type systems must be continually adapted else they become progressively less satisfactory in use
II 1974	Increasing Complexity	As an *E*-type system is evolved its complexity increases unless work is done to maintain or reduce it
III 1974	Self Regulation	Global *E*-type system evolution processes are self-regulating
IV 1978	Conservation of Organisational Stability	Average global activity rate in an *E*-type process tends to remain constant over stages or segments of system evolution

V 1978	Conservation of Familiarity	In general, the incremental growth and long term growth rate of *E*-type systems tend to decline
VI 1991	Continuing Growth	The functional capability of *E*-type systems must be continually increased to maintain user satisfaction over the system lifetime
VII 1996	Declining Quality	Unless rigorously adapted to take into account changes in the operational environment, the quality of *E*-type systems will appear to be declining
VIII 1996	Feedback System (Recognised 1971, formulated 1996)	*E*-type evolution processes are multi-level, multi-loop, multi-agent feedback systems

Table. MML.1 : The Laws of Software Evolution (as currently stated)

The laws were numbered in the order of their formulation and have been refined over the years as understanding of the phenomena encapsulated in them has advanced [leh74,85,fea00b]. They were termed *laws* because they reflect human social, organisational and managerial forces and because they were believed to encapsulate behaviour and phenomena that are, in general, outside the purview of those concerned with day to day development, the direct application of methods, tools and techniques to evolve the software. In general, FEAST results have increased confidence in six of the eight laws. Law VII was not investigated due to lack of appropriate data, though it is supported by theoretical reasoning [leh89,fea00a]. Law IV reflects present understanding and requires further investigation [leh98,raj00]. Based on the laws and on the behaviour they imply, a number of recommendations for long term evolution process planning and management have recently been compiled [leh00]. Readers interested in this and further details regarding the laws are referred to the project web site [fea00a].

The eighth law provides an explicit statement of the feedback system nature of software evolution processes. Its formulation as a law was based on many years of observations of the programming process. As discussed above, the long term growth patterns observed in several systems can be interpreted in terms of compensating positive and negative feedback loops. Positive feedback influences such as pressures to increase functional power to improve sales or augment the customer base, on the other hand, tend to be constrained by the global software process capacity to implement and assimilate change. Given that, in general, managers do not directly or explicitly control system growth rate as such, one may say that observed growth trends are, at least in part, the result of *controls* imposed by counteracting positive and negative feedback forces.

Additional support for the law has been provided by simulation modelling of software processes. New insight, in particular, has been provided by a number of *system dynamics* [for61] models built by the FEAST group. These include feedback influences that are believed to be dominant [wer98,cha99,kah00]. Feedback influences tend also to be reflected in simulation-based models of industrial processes presented in the literature, such as those in the ProSim workshops [pro00].

Role and Impact of Feedback in *E*-type Processes

As a part of its goals, improvement of the evolution process must seek to achieve systematic and incremental changes to the process and its constituents to obtain gains in productivity, timeliness, product quality and so on. It must also seek to compensate for deterioration of these attributes as a result of, for example, increasing software complexity, loss of key personnel, deterioration of architectural integrity and so on. As mentioned above, the process is in fact a complex feedback system. Difficulties in achieving improvement will, therefore, be experienced if the feedback forces are not adequately considered. Without going into details as to what type of behaviour in the attributes to see of a complex feedback process such as that depicted in figure MML.3 one should expect, one may postulate some immediate observations in the context of process improvement. For example, local process improvements may not be observable from outside the system, that is, they may not be reflected in the overall (global) process or its product. Process changes may have a counterintuitive effect on global process behaviour. The effectiveness of methods, tools and techniques as established in a small setting or *in vitro* conditions, may not be apparent when such methods, tools and techniques are deployed in a large scale or *in vivo* situation. Process-local optimisation may not lead to global optimisation and so on. Major process improvement requires, *inter alia*, appropriate consideration, or adjustment, of sub-processes and feedback-related dynamic influences. Moreover, in general, outer loops that cannot, be strongly, influenced by individual developers, but may have a dominant influence on process outcome. This leads to an overwhelming set of problems whose recognition opens up new approaches to improvement.

Black-box and White-box Modelling and Models

The starting point of the FEAST investigation was the application of the accepted scientific method (see figure MML.1) to the investigation of the role and impact of feedback in the software process, with particular emphasis on process modelling. It recognised the fact that understanding the dynamic influences in a complex feedback process requires appropriate quantitative models [wid89].
In seeking process understanding, the approach has, to date, involved construction of quantitative models. These aim to reflect the long-term behaviour of attributes of such processes and their product. In so doing, FEAST distinguishes between two types of quantitative models: *black-box* and *white-box*. Though different, these are related, even, in some respects, complementary. Black box models encapsulate relationships between attributes. They primarily reflect structure in and behaviour of metrics and hence process inputs and outputs, as sketched in Figure MML.4. The graph formed by solid line rectangles (small) connected by arrows represents a view of a software process, with the rectangles representing, for example, activities that generate or update process documents and other artefacts (including the source code). The arrows represent dependency relationships between such activities. The dashed line shadowed rectangle is intended to suggest that in black-box modelling only behaviour seen from outside the rectangle is of interest. The arrow and the smaller box at the bottom of the figure indicate that the modelling activity follows a top-down approach with successive refinement. To the right of the figure, input-output relationships and input-output behaviour over time or pseudo-time (release sequence numbers RSN [cox66]) and typical relationships of interest are indicated.

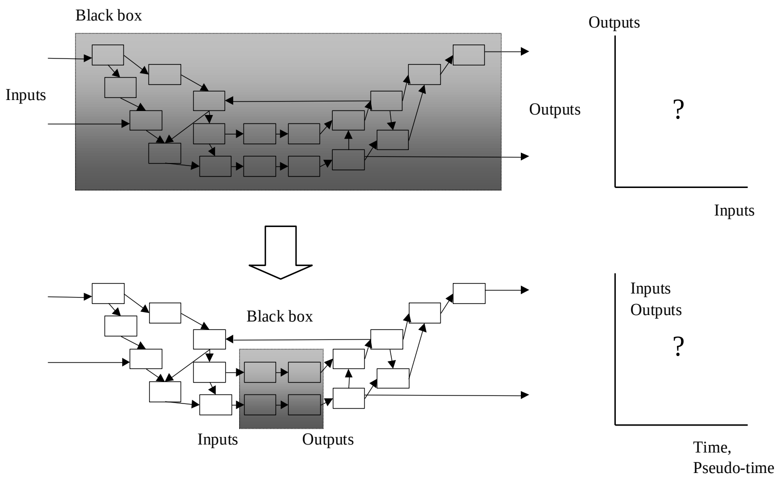


Fig. MML.4 : Black-Box Models in FEAST

White box models reflect process structure, its major constituents, sub-processes and relevant details of their interactions. The goal is to convey insight into the internals of the process modelled. Figure MML.5 is analogous to Figure MML.4 but represents white-box modelling activity. The dashed line rectangle, here, encompasses the part of the process being modelled. The right hand side of Fig. MML.5, is intended to suggest that, in principle, *any* attribute not only inputs and outputs, may be of interest when developing and executing white-box models.

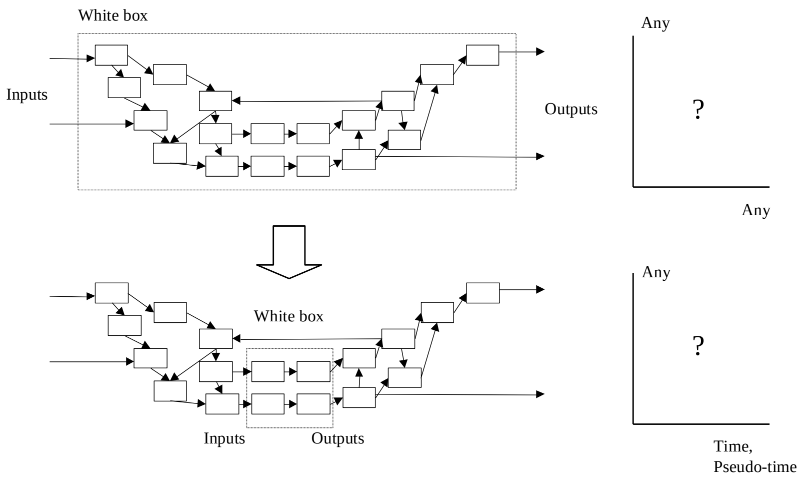


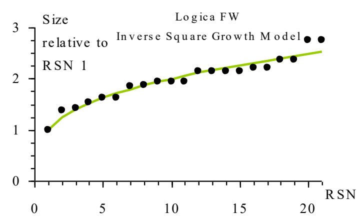
Fig. MML.5 : White-Box Models in FEAST

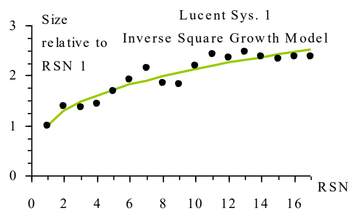
In the wider software community, the two modelling approaches are generally pursued separately. Examples of black-box modelling as defined here are to be found in the software metrics literature. Activity leading to models analogous to our white-box models has been primarily pursued in the software process simulation modelling community [e.g., pro00]. The FEAST modelling approach and studies, to date have differed in several ways. They have, for example:

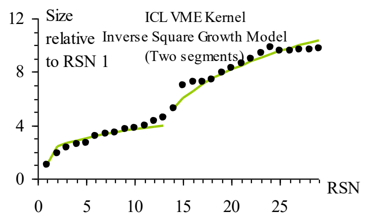
- aimed at process *understanding* and at *identification* of *feedback* loops by pursuing models with a degree of explanatory power
- focused on modelling long-term behaviour, with time units of releases or years
- followed a top-down approach that starts with a small number of variables, and is then followed by successive model refinement to add detail as understanding progresses [zur67]
- reflected primarily software evolution attributes, based on modules counts, when the majority of the work in the literature concentrates on lines of code or function points based metrics
- above all, the FEAST has sought to combine black-box and white-box modelling

The models reflected in the figures that follow illustrate this approach.

The first, Figure MML.6, displays the results of fitting an inverse square model of growth over release sequence numbers (RSN) to data from a number of systems studied in FEAST. The model contains only one parameter E. The prediction error, measured by the mean absolute error (mae) is 8 percent or less, for the systems in the table contained in Fig. MML.6. For two of the systems studied to date, partitioning the interval and fitting separate models over successive intervals improved the fit. Further details on the inverse square models are available in the FEAST literature [tur96,fea00a]. The reader seeking further detail or guidelines in building black box models is referred to [ram00].







$S_i = S_{i-1} + E/(S_{i-1})^2$ i= 2,...,n

S_i = predicted size of system at rsn i

E = average E_i

$E_i = (y_i-y_1)_{k=1}S^{i-1}(1/y_k^2)$ i = 2,...,n

y_i = actual size of system at rsn i

System	Model	% MAE	St. Dev. of % Residuals
System X First 7 years	Inv. Sq.	4.0	5.3
System X Last 4 years	Inv.Sq.	1.7	2.2
	Linear	1.5	1.9
FW	Inv. Sq.	3.6	4.8
Lucent Sys 1	Inv. Sq.	6.0	7.2
Lucent Sys 2 - rsn 1- 5	Linear	5.1	7.6
Lucent Sys 2 - rsn 7 - 14	Inv.Sq	1.9	2.6
	Linear	1.4	1.8
OS/360-370	Linear	4.7	5.9
VME Kernel - rsn 1 - 13	Inv. Sq.	8.0	11.4
VME Kernel - rsn 14 - 29	Inv. Sq.	3.7	5.1

Mean Abs. Perc. Error %MAE $= 100 . n^{-1} \Sigma(|S_i - y_i|/y_i)$ i= 1,...,n

% Residual$_i = 100 .(S_i - y_i)/y_i$

Fig. MML.6 : Example of Black-Box Modelling in FEAST

Figure MML.7a presents a system dynamics [for61] model that reflects part of the most recent FEAST/2 work. A detailed discussion of this model and the underlying concepts is given in [kah00]. Further guidelines are given in [ram00].

An example of the output of this model is shown in fig. MML.7b. This figure shows the simulated growth of the system to month 180 compared with actual growth under alternative management policies. These policies relate to

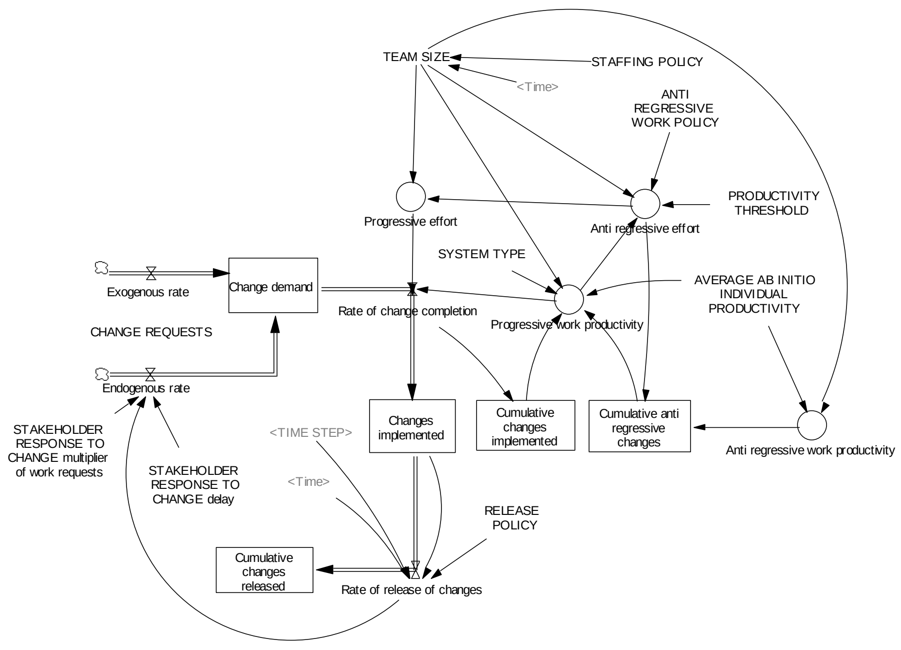

TEAM SIZE
STAFFING POLICY
<Time>
ANTI REGRESSIVE WORK POLICY
PRODUCTIVITY THRESHOLD
Progressive effort
Anti regressive effort
SYSTEM TYPE
AVERAGE AB INITIO INDIVIDUAL PRODUCTIVITY
Exogenous rate
Change demand
Rate of change completion
Progressive work productivity
CHANGE REQUESTS
Endogenous rate
STAKEHOLDER RESPONSE TO CHANGE multiplier of work requests
<TIME STEP>
Changes implemented
Cumulative changes implemented
Cumulative anti regressive changes
Anti regressive work productivity
STAKEHOLDER RESPONSE TO CHANGE delay
<Time>
RELEASE POLICY
Cumulative changes released
Rate of release of changes

Fig. MML.7a : Example of White-Box Modelling in FEAST

the portion of resources applied to anti regressive [leh74] work such as complexity control. AR = 0.3 indicates that 30 percent of resources are devoted to this activity, AR = 0.0 indicates that no resources are made available to it.

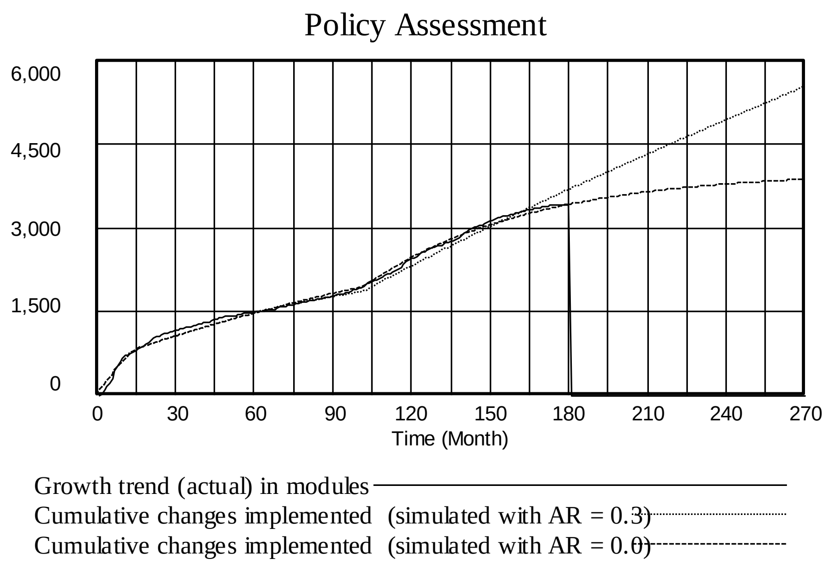


Fig. MML.7b : An Example of the Output of the Model in Fig. MML.7a.

Iterative Modelling for Process Improvement

As a system evolves one must expect to observe the consequences of increasing constraints as the software ages. This may be due to the successive superposition of changes, to the orthogonality of such changes, to increasing remoteness from the original architecture, to growing complexity, personnel turnover and so on. Recognition of such factors provides opportunities for process improvement by investigating and improving appropriate aspects of the process. Such improvement needs to occur over an extended period of time over several releases. Evolution processes tend to be, by their very nature, cyclic with each cycle generating a new release of the operational software. One takes advantage of this by combining black-box and white-box modelling in an iterative fashion [ram00] to exploit the complementary features of the two classes of models to achieve increased understanding of a given evolution process and to support its improvement by means of quantitative models. The diagram of figure MML.8 represents the proposed iterative process. It represents a meta-model for a measurement and modelling driven improvement process. Two basic activities are highlighted: monitoring of key global process attributes by means of black-box modelling and the identification of potential improvements by means of white-box modelling. In principle, models should be updated at every cycle of the iterative evolution process to reflect changes in it.

Final Remarks

This paper has presented a short summary of the concepts and process modelling approach underpinning the FEAST investigation with the hope that they may be of benefit to both researchers and practitioners in industry and academia. It reports on the work of a single team. It has been clear since the start of the FEAST investigation that achievement of a better understanding of the role and impact of feedback in software processes requires an interdisciplinary approach. Such approach would take into account, for example, business, social, organisational, managerial, cognitive and other aspects of the global process. The investigation would need the co-operative work of many groups. In view of the peculiar characteristics of the software process, a simple transfer of results from related fields (e.g., from management and organisational theory) is unlikely to provide appropriate answers.

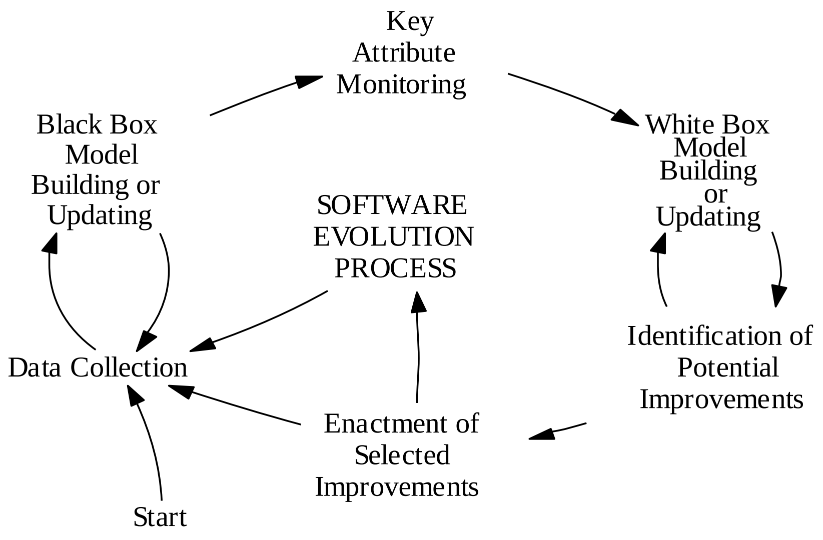

Key Attribute Monitoring
White Box Model Building or Updating
Identification of Potential Improvements
Enactment of Selected Improvements
Data Collection
Black Box Model Building or Updating
SOFTWARE EVOLUTION PROCESS
Start

Fig. MML.8 : Iterative Modelling

Software processes display and share characteristics with industrial production that transform physical entities. The system dynamics of these have been studied by, for example, Forrester and co-workers since the sixties [for61]. In many respects they also resemble invention, design and R&D processes whose dynamics are not so simple to grasp or to model. Additionally, when compared to other artificial artefacts (in Simon's sense [sim69]), software evolution occurs at a faster rate. Software changes are motivated by organisational and technological changes and frequently triggered and accelerated by the introduction of new releases of the software in the operational domain [leh85]. In this sense, software evolution can be considered the *Drosophila Melanogaster*, the *fruit fly* of artificial processes.

Acknowledgements

We gratefully acknowledge the collaboration of Dr. Paul Wernick for his contribution in the building of the system dynamics model presented and as a member of the FEAST/1 team . Many thanks are due to Ms. Siew F. Lim for her continuing support. Financial support from the UK EPSRC, grant GR/M44101 (FEAST/2) and GR/N02412 (SVFs) is gratefully acknowledged.

References - references identified with an * are reprinted in [leh85]

[bel72]* Belady LA, Lehman MM, *An Introduction to Program Growth Dynamics*, in Stat. Comp. Perf. Ev., W. Freiburger (ed.), Ac. Press, NY, 1972, pp. 503-511

[cha99] Chatters BW, Lehman MM, Ramil JF, Wernick P, *Modelling a Software Evolution Process*, ProSim'99, Softw. Process Modelling and Simulation Workshop, Silver Falls, Oregon, 28-30 Jun. 1999, also as *Modelling a Long Term Software Evolution Process* in J. of Softw. Proc.: Improvement and Practice, 2000, v. 5, iss. 2/3, Jul. 2000, pps. 95-102

[cox66] Cox DR, Lewis PAW, *The Statistical Analysis of Series of Events*, Methuen, London, 1966

[fea94,95] *Preprints of Three FEAST Workshops*, Lehman MM (ed.), Dept. of Comp., Imp. Col., 1994/5 [fea00a] FEAST, *Feedback, Evolution and Software Technology*, web site: http://www-dse.doc.ic.ac.uk/~mml/feast

[fea00b] *Preprints of FEAST 2000 International Workshop on Feedback and Evolution in Software and Business Process*, Ramil JF (ed.), Dept. of Comp., Imp. Col., London, 10 - 12, Jul. 2000, 124 pp. http://www-dse.doc.ic.ac.uk/~mml/f2000

[for61] Forrester, J., *Industrial Dynamics*, The MIT Press, 1961.

[kah00] Kahen G, Lehman MM, Ramil JF, Wernick PD, *Dynamic Modelling in the Investigation of Policies for E-type Software Evolution*, ProSim 2000, 12 - 14 Jul. 2000, London, UK

[leh69]* Lehman MM, *The Programming Process*, IBM Research Report RC 2722, IBM Research Centre, Yorktown Heights, NY, Sept. 1969

[leh74]* Lehman MM, *Programs, Cities, Students, Limits to Growth?*, Inaugural Lecture, in Imperial College of Science and Technology Inaugural Lecture Series, Vol. 9, 1970, 1974, pp. 211-229. Also in Programming Methodology, (D. Gries. ed.), Springer Verlag, 1978, pp. 42-62

[leh85] Lehman MM, Belady LA, *Program Evolution—Processes of Software Change*, Academic Press, London, 1985.

[leh89] Lehman MM, *Uncertainty in Computer Application*, Comm. of the ACM, Vol. 33, No. 5, May 1990, pp. 584-586

[leh90] Lehman MM, *Uncertainty in Computer Application*, Technical Letter, Comm. of the ACM, vol. 33, no. 5, pp. 584, May 1990

[leh94] Lehman MM, *Feedback in the Software Evolution Process*, Keynote Address, CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics. Dublin, 7-9 Sept. 1994, Workshop Proc., Information and Software Technology, sp. is. on Software Maintenance, v. 38, n. 11, 1996, Elsevier, 1996, pp. 681 - 686

[leh98] Lehman MM, Perry DE and Ramil JF, *On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution*, Proc. Metrics'98, Bethesda, MD, 20-21 Nov. 1998, pp. 84-88

[leh00] Lehman MM, *Rules and Tools for Software Evolution Planning and Management*, FEAST 2000 Workshop, 10 - 12 July 2000, Imp. Col.

[pro00] Prosim 98, 99 and 2000, *Software Process Simulation and Modeling Workshops*, http://www.prosim.org

[ram00] Ramil JF, Lehman MM, Kahen G, *The FEAST Approach to Quantitative Process Modelling of Software Evolution Processes*, Proc. PROFES'2000 2nd International Conf. on Product Focused Software Process Improvement, Oulu, Finland, 20-22 Jun. 2000, in Frank Bomarius and Markku Oivo (eds.) LNCS 1840, Springer, Berlin, 2000, pp. 311-325

[raj00] Rajlich VT and Bennet KH, *A Staged Model for the Software Life Cycle*, Computer, July 2000, pp. 66 - 71

[sim96] Simon HA, *The Sciences of the Artificial*, 3rd. ed. The MIT Press, Cambridge, MA, 1996, 231 pp, first pub. in 1969

[tur96] Turski WM, *A Reference Model for the Smooth Growth of Software Systems*, IEEE Trans. Softw. Eng., v. 22, n. 8, Aug. 1996, pp. 599 - 600

[wer98] Wernick P and Lehman MM, *Software Process White Box Modelling for FEAST/1*, ProSim '98 Workshop, Silver Falls, OR, 23 Jun. 1998. As a revised version in J. of Sys. and Softw., v. 46, n. 2/3, 15 Apr. 1999

[wid89] Widman LE and Loparo KA, *Artificial Intelligence, Simulation, and Modeling - A Critical Survey*, in Widman LE, Loparo KA and Nielsen NR (eds.), Artificial Intelligence, Simulation and Modeling, Wiley, NY, 1989

[zur67] Zurcher FW and Randell B, *Iterative Multi-Level Modelling - A Methodology for Computer System Design*, IBM Res. Rep. RC 1938, Nov. 1967, IBM Res. Centre, Yorktown Heights, NY 10594. Also in Information Processing 67, Proc. IFIP Congr. 1968, Edinburgh, Aug. 1968, pp. D138 - 142

Intranet as a vehicle and a platform for Process Improvement

Roberto Delmiglio - Sergio Scotto di Vettimo
Siemens Information and Communication Networks S.p.A. – Communication on Air – Mobile Radio, Milano - ITALY
Francesco Basili - Gualtiero Bazzana - Giannino Zontini
Onion S.p.A., Brescia - ITALY

Introduction

This paper describes the Intranet structure, applications, methods and tools in use at Siemens Information and Communication Networks SpA – Information and Communication Mobile - Communication on Air - Research and Development department (Siemens ICN ICM CA R); the Intranet project concerns with a lasting process improvement activity focused on the software development related with mobile telephony applications. The paper is organized around the following topics:

- a general description of the Internet and Intranet technologies and applications;
- the Software Process Improvement (SPI) activities within which the Intranet project has been developed;
- a detailed description of the most important developed services for software process management and improvement and for project management;
- a general appraisal of lessons learnt and future goals.

Internet and Intranet

The application and use of Internet standards and technology within an organization is termed Intranet. The Internet itself is difficult to describe. Some refers to it as regulated anarchy, but accurate. The Internet is best described as a non-proprietary standard for connecting information networks, “a network of networks”.
The Intranet is perhaps the greatest technology improvement for organizations since the direct dial telephone. In simple terms, the Intranet is the descriptive term being used for the implementation of Internet technologies within a corporate organization, rather than for external connection to the global Internet. This implementation is performed in such a way as to transparently deliver the immense

informational resources of an organization to each individual's desktop with minimal cost, time and effort. The main technology components of the Internet are:

Communications Protocol
The ability to connect and communicate between networks and individual desktop devices.

File Transfer
The ability to transfer files between point-to-point locations.

Mail
The ability to provide direct point-to-point communication between individuals or groups.

Web Browsing
The ability to provide access to information on a one to many basis, on demand.

Terminal Emulation
The ability to access existing infrastructure applications.

User Interfaces
The ability to deliver the increasing technical complexity to the desktop in a transparent, seamless and intuitive manner.

Internet technologies are actually extremely well suited for developing internal corporate information systems - the Intranets. In fact, Internet technologies are much more relevant and exploitable within a local LAN, right now, than over much slower, dial-up access routes associated with typical home-access to the Internet.

Within the early Intranet adopters, the application of this hot technology is being typically used as follows:

1. *Publishing corporate documents*
Along with oft-mentioned human resource guides, these documents can include newsletters, annual reports, maps, company facilities, price lists, product information literature, and any document which is of value within the corporate entity. This is one area where significant cost control can be achieved as well as much more efficient, timely and accurate communication across the entire corporate organization.

2. *Access into searchable directories*
Rapid access to corporate phone books and the like. This data can be mirrored at a Web site or, via CGI scripts, the Web server can serve as a gateway to back-end pre-existing or new applications. This means that, using the same standard access mechanisms, information can be made more widely available and in a simpler manner.

3. *Corporate/Department/Individual pages*
As cultures change within organizations to the point where even each department moves towards their own individual mission statements, the Internet technology provides the ideal medium to communicate current information to the Department or Individual. Powerful search engines provide the means for people to find the group or individual who has the answers to the continuous questions which arise in the normal day-to-day course of doing business.

4. *Simple GroupWare applications*
With HTML forms support, sites can provide sign-up sheets, surveys and simple scheduling. As the Intranet technologies continue to evolve, the press have been positioning the technologies as alternatives to major GroupWare applications (e.g. Lotus Notes) to such a point that this type of rhetoric only serves to cause confusion as to the appropriateness of each area of technology. As always, it is not black and white. The Intranet technology can be used to complement or as an alternative to such GroupWare products. It all becomes a matter of scale, cost, timescale, openness and taste.

5. *Software distribution*
Internal Administrators can use the Intranet to deliver software and up-dates 'on-demand' to users across the corporate network. This will continue to gain momentum as new technologies such as Java become more widely available from Sun, which will allow the creation and transparent distribution of

objects on-demand rather than just data or applications.

6. *Mail*
Although email has been seen as being a ‘killer application’, its uses have tended to be limited and over-complicated. With the move to the use of Intranet mail products with standard and simple methods for attachment of documents, sound, vision and other multimedia between individuals, mail is about to be pushed further forward as a simple, de facto communications method. Mail is essentially individual to individual, or individual to small group, communication. With the emergence of Web technology, there are now better and more appropriate tools for one-to-many communication which historically is where mail systems have been over-burdened and over-burdening to the point of reducing their effectiveness.

7. *User Interface*
The Intranet technology is evolving so rapidly that the tools available, in particular HTML, can be used to dramatically change the way we interface with systems. There will be a significant debate, which will shortly hit the streets, which will question ‘Graphical User Interfaces (GUIs) vs. End User Comfortable Interfaces’ . At the beginning of the 90’s, the industry was deluged with the increases in productivity from GUI’s. However, no-one anticipated the converse loss of productivity by normal business users being able to access the wealth of functionality provided by Windows. The GUI has been defined by Microsoft as an iconic desktop. But, although this might be what technicians like - and like to believe it’s what users like - it is definitely not the interface that most business people are comfortable with. With HTML you can build an ‘End User Comfortable Interface’ which is only limited by the creator’s imagination. The beauty about using Intranet technologies for this is that it is so simple. Hitting a hyperlink from HTML does not necessarily take you to another page - it could ring an alarm, run a year end procedure or anything that a computer action can do. Microsoft’s Windows 3.x and Windows 95 created tremendous volumes of functionality, but individuals probably only need 5% of the total functionality. The other 95% causes support pain, headaches and disruption. Now, with the Intranet tools, you can paint reality in HTML and make an in-context and uniform front-end to all computer-based resources. In doing so, not only can you create interfaces that users can use and appreciate , you can also remove the 95% functionality and access to elements that specific users don’t need - getting rid of most of your headaches in one sweep.

Intranet technologies provide the tools, standards and new approaches for meeting the problems of today’s business world. The beauty about most of these technologies is that they are simple and, in their simple elegance, phenomenal power can be unleashed. Due to the fact that these technologies are still moving from adolescence to maturity, there are many rough edges. The route ahead, however, is being well-defined and the new generations of Intranet products designed specifically for corporate use will address these. Communication is the key to business success. Exploitation of the Intranet is the key to effective and efficient communications [1].

Intranet and Software Process Improvement

The Intranet project described in this paper is strongly involved within the Software Process Improvement (SPI) program that the Mobile Networks Research and Development department is undergoing since 1995, with the following high-level goals:

- to optimise the predictability of schedules;
- to further enhance product quality;
- to raise the availability and usability of documentation;
- to better the tool support;
- to keep/ increase productivity levels.

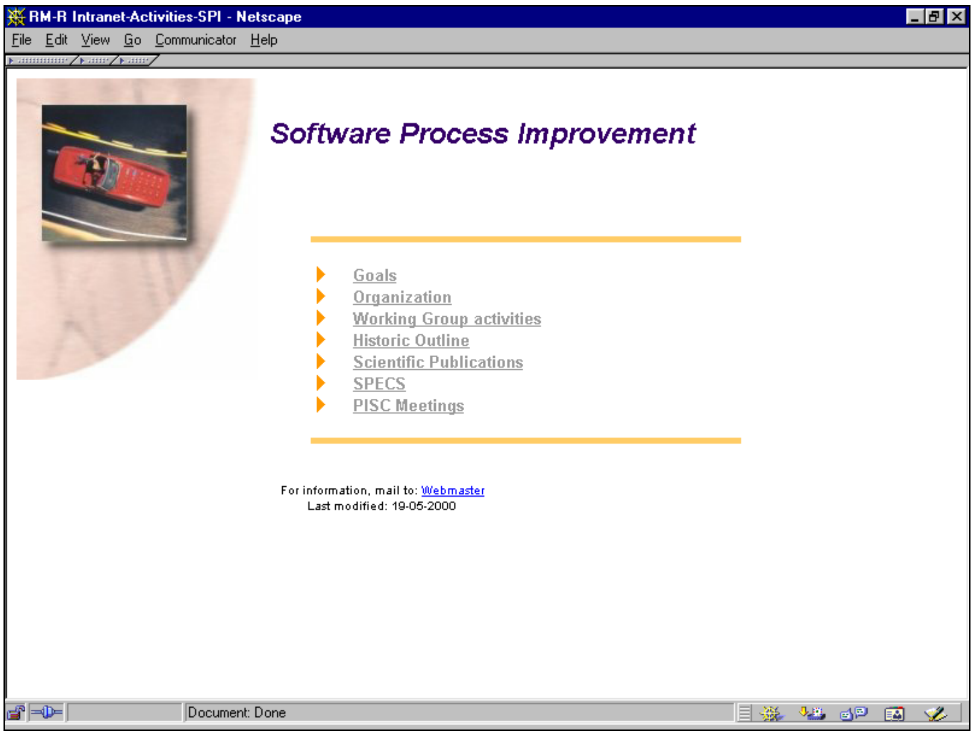
Mobile Networks Research and Development department is strongly committed to Software Process Improvement that is felt as a major leverage to increase the company capabilities; this is motivated by: the high world-wide competitiveness in the target domain, the increasing complexity of the software embedded in the delivered products and systems, the fact that projects are developed on an international multi-site basis and the increasing requirements from customers that are more and more demanding on software process maturity and stability.

The SPI program started in May-June 1995 with a *Software Process Assessment* conducted by Siemens Central Research, Application Centre Software. The assessment highlighted a good maturity level for the software producing unit, given that SPI was already an established practice. The SPI Program is intended as a continuous effort, handled with a management-by-objective approach with milestones and quantitative results.

In order to ensure its success, the PI (Process Improvement) Project has been organized as:

- a *PI Steering Committee* (referred in the following as "*PISC*"), chaired by the R&D Director and including all the managers reporting to him. The aim of this board is to define priorities, assign resources, solve problems and track the success of the initiative;
- a *PI Project Office* (called "*PIPO*" and equivalent to a SEPG), composed of a few experts, having the goal of planning/ tracking the project, giving technical guidance and harmonizing/ deploying the outcomes of the Working Groups; the PIPO has also the duty to organize the so-called "accompanying actions", namely: training, dissemination and quantitative measurement.
- a number of *Working Groups*, composed of technical representatives from the various projects involved and dealing with improvement actions.

Working groups are subdivided in two areas: the former with the goal to innovate that is to define/ enhance software engineering practices, the latter to deploy that is to apply defined practices, into development projects.
Within the Intranet the Software Process Improvement Program is fully described and managed in terms of its goals, organization, historic outline and current activities defined for each workgroup; all the related scientific publications and meetings minute are collected and available to any user (Fig. FB.1).


RM-R Intranet-Activities-SPI - Netscape
File Edit View Go Communicator Help
Document: Done

Software Process Improvement

- Goals
- Organization
- Working Group activities
- Historic Outline
- Scientific Publications
- SPECS
- PISC Meetings

For information, mail to: Webmaster
Last modified: 19-05-2000

Fig. FB.1 Software Process Improvement: the Intranet page

The Intranet developed services represent a necessary vehicle to improve both the software process management, the software development process and the project management. Intranet is also used as a channel to deploy most of the improvement actions established by the Process Improvement Steering Committee.

The Intranet at Siemens ICN ICM CA R

The Intranet project goals defined for the Siemens ICN ICM CA R are:

- to provide web based communication services to the Research and Development (R&D) staff;
- to provide info about R&D activities and products;
- to give a common reference on the Intranet for the whole R&D;
- to provide the basis for the most common and useful services for R&D.

The set-up approach was first to specify roles and responsibilities within the project; a responsibility matrix has been defined for the following roles related with various parts of the Siemens ICN ICM CA R Intranet, namely:

- *Service developer*: this responsibility is associated with the design/ development/ maintenance of added value Intranet services which imply programming rather than simple information publishing;
- *Info provider*: this responsibility is associated with the provision of up-to-date information so that the WWW is always aligned with the current state of the art;
- *Reviewer*: this responsibility is associated with the review of the contents proposed by the info provider, both from the point of view of its semantic correctness and of its suitability with respect to the purpose of the Intranet;

- *Publisher*: this responsibility is associated with the management of information on the WWW server, so that it can be visible to the users.

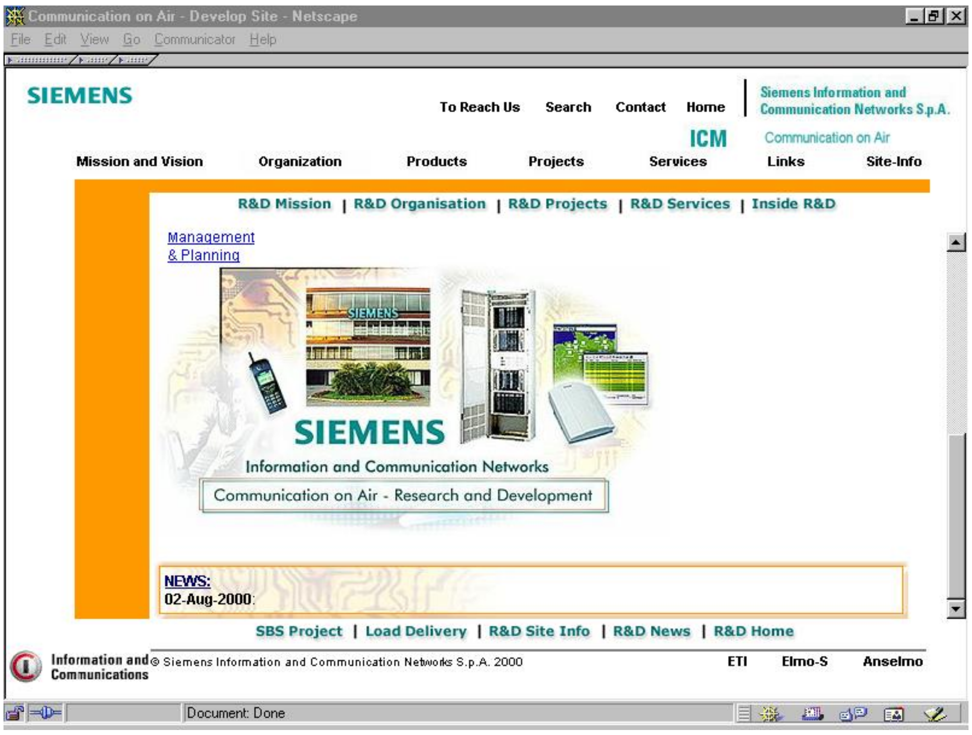

Communication on Air - Develop Site - Netscape
File Edit View Go Communicator Help
SIEMENS
To Reach Us
Search
Contact
Home
Siemens Information and Communication Networks S.p.A.
ICM
Communication on Air
Mission and Vision
Organization
Products
Projects
Services
Links
Site-Info
R&D Mission | R&D Organisation | R&D Projects | R&D Services | Inside R&D
Management & Planning
SIEMENS
Information and Communication Networks
Communication on Air - Research and Development
NEWS:
02-Aug-2000
SBS Project | Load Delivery | R&D Site Info | R&D News | R&D Home
Information and Communications
© Siemens Information and Communication Networks S.p.A. 2000
ETI
Elmo-S
Anselmo
Document: Done

Fig. FB.2 The Siemens ICN ICM CA R intranet home-page

Figure FB.2 shows the Siemens ICN ICM CA R Intranet home page; starting from the home page it is possible to reach all the services and available documentation in terms of company mission, vision and organization, products, services and links to the other corporate internet and intranet sites.

Referring to the services section, the Intranet offers links to several web-applications related to the following topics:

- *Configuration management* section provides a web-client application for accessing the company configuration management environment;
- *Telecommunications Standards:* a repository of all the documents that are related to the defined GSM and UMTS standards;
- *Training on line*: this section makes available all the updated documentation used during the employees training sessions;
- *Technical documentation*: provides a lot of information concerning tools, programming languages, operating systems, internet technologies, network management etc …
- *Mailing groups*; a number of mailing lists has been defined in order to simplify sending E-mails to large groups of people. A user is able to request the creation of a new mailing list, to add or remove a user. Each list is composed of the Mail Alias field that contains the mailbox to click on in order to open a mail form to the group; the Description field that contains the explanation of the list; the Owner field that corresponds to the people to contact for updating the mailing list; the List field that can be clicked to get details of the people that currently belongs to the specified mailing list.
- *Support tools*: starting from this page is possible to run several tools and utilities. The most important one will be described in the following.
- *Information, statistics* and technical structure of the network.

Intranet Case Studies

MEDAL (MEasurement DAta Library)

MEDAL (MEasurement DAta Library) is a web-application that supports the software process management and improvement providing a repository for collection, analysis and reporting of quantitative data from software development projects. MEDAL manages all the software related quality indicators and detailed historical data about scheduling, lines of code, delta lines of code, planned and actual effort and faults (see Fig. FB.3).

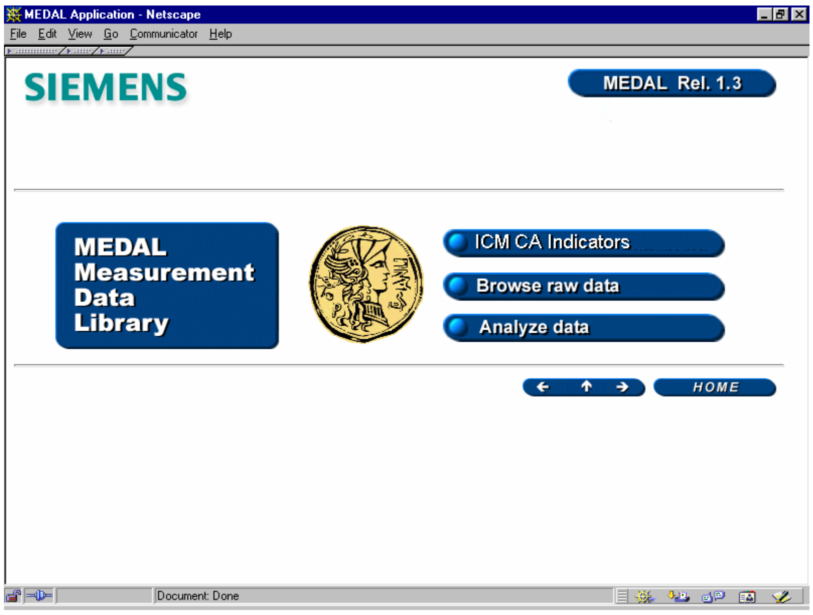

MEDAL Application - Netscape
File Edit View Go Communicator Help
SIEMENS
MEDAL Rel. 1.3
MEDAL Measurement Data Library
ICM CA Indicators
Browse raw data
Analyze data
HOME
Document: Done

Fig. FB.3 The MEDAL home page

A careful security management through usage profiles is established; depending on the given permissions a MEDAL user can access to the service using a normal web browser performing one of the following actions:

- View the data
- Input new data
- Administrate the system and maintenance the data tables.

Using the *historical frozen values* related section it is possible to retrieve the frozen values of a specific indicator across the various Releases for which it is available. Graphic charts can be extracted for a specific Release as well as for all Releases; a special feature has been developed for Siemens Base Station (SBS) Project, providing also the possibility of a chart with evidence of values for all Network Elements; it can be produced by clicking on ALL after the first search has been completed.

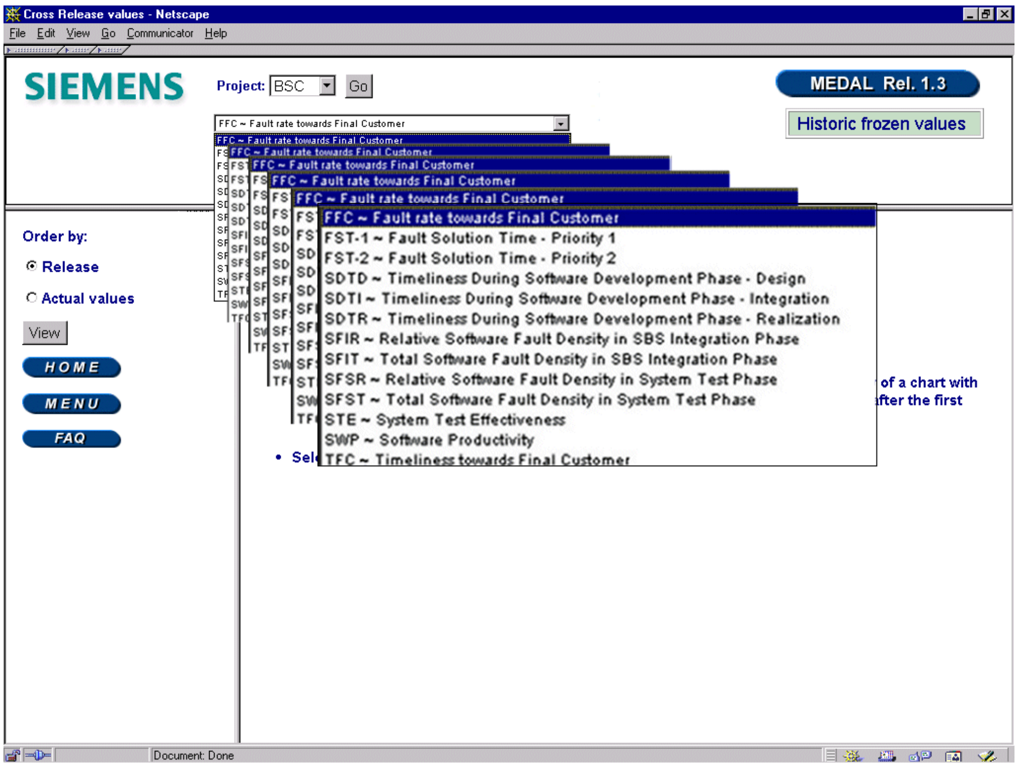
The Software Quality Indicators can be analysed using two different modalities: in a Release oriented mode or an Indicator oriented mode.

The Release oriented mode allows the user to retrieve the values of all indicators pertaining to a specific Release of a Project, giving details of the monthly values; indicators which are collected at baselines, are reported only for the specific months in which an applicable milestone has been reached; graphic charts can be extracted for all indicators with evidence of Target and actual values.

The Indicators oriented mode allows the user to retrieve the values of a specified indicator for a Project, giving details of the monthly values (for applicable periods) in all the Releases for which the

Indicator has been computed. Months are reported as absolute numbers starting from one, with indication of the first month for each Release. Also in this section graphic charts can be extracted for a specific Release as well as for all Releases.

As mentioned before, MEDAL is a repository for collecting, analysis and reporting of quantitative data from software development projects; some of the managed software related quality indicators are shown in the next figure.



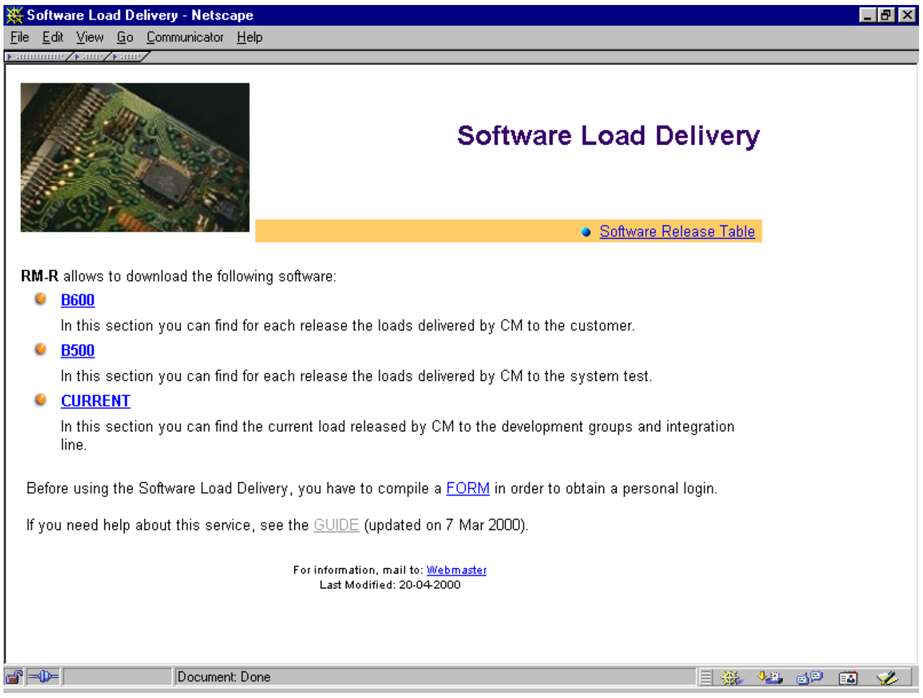
FFC ~ Fault rate towards Final Customer
FST-1 ~ Fault Solution Time - Priority 1
FST-2 ~ Fault Solution Time - Priority 2
SDTD ~ Timeliness During Software Development Phase - Design
SDTI ~ Timeliness During Software Development Phase - Integration
SDTR ~ Timeliness During Software Development Phase - Realization
SFIR ~ Relative Software Fault Density in SBS Integration Phase
SFIT ~ Total Software Fault Density in SBS Integration Phase
SFSR ~ Relative Software Fault Density in System Test Phase
SFST ~ Total Software Fault Density in System Test Phase
STE ~ System Test Effectiveness
SWP ~ Software Productivity
TFC ~ Timeliness towards Final Customer

Fig. FB.4 Historic Frozen values

SLIVER (Software deLIVERy)

The “SBS Software Delivery” environment (see Fig. FB.5) is an Intranet service thought with the aim to make easier the distribution of software loads to users. Using this service, users are able to build tapes, disks, and CD-ROM by themselves, at their sites, whenever they need, *avoiding the distribution of physical devices* by the Configuration Management group.
Typical users of this environment are development teams, Integration Lines, System Test and Customer Service departments at Siemens ICN S.p.A. and Siemens AG that need copies of the official software loads. They access the environment via a standard browser (Netscape Navigator or Microsoft Internet Explorer) and can obtain exactly the software needed. The service is granted under access control restrictions.

Via the Software Load Delivery environment, the user has access to the executable loads of the Base Station Sub-System. Depending on the selected network element, the user can download software on the desired media, install software directly from the Intranet on his/her workstation or upgrade the software installation.


Software Load Delivery - Netscape
File Edit View Go Communicator Help

Software Load Delivery

● Software Release Table

RM-R allows to download the following software:

- B600

  In this section you can find for each release the loads delivered by CM to the customer.
- B500

  In this section you can find for each release the loads delivered by CM to the system test.
- CURRENT

  In this section you can find the current load released by CM to the development groups and integration line.

Before using the Software Load Delivery, you have to compile a FORM in order to obtain a personal login.

If you need help about this service, see the GUIDE (updated on 7 Mar 2000).

For information, mail to: Webmaster
Last Modified: 20-04-2000

Document: Done

Fig. FB.5 Software Load Delivery

It is possible todownload software on three different kinds of media:

- tape: DAT tape in Unix tar format;
- CD-ROM: recordable CD-ROM.
- floppy disk: 1.44 MB (3.5" HD) DOS formatted floppy disks;

The Software Load Delivery environment is divided in three areas, corresponding to the baselines of the project life cycle: a section contains for each release the official loads delivered by Configuration Management mainly for customer service usage; another section is used to store every load delivered by Configuration Management mainly for system test usage; the last section is used to store the current work-in-progress load released by Configuration Management mainly for development groups and integration lines.

WATCH (Web Access To CM Hub)

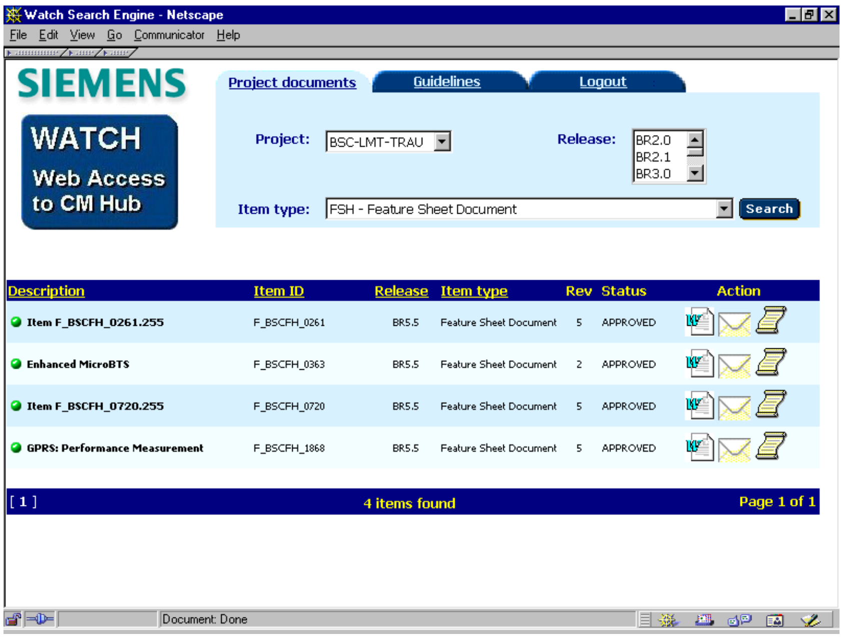
The WATCH web-service provides easy access to the Siemens ICN ICM CA R Configuration Management (CM) environment enabling browsing facilities characterised by an intuitive user interface.
No need for knowledge of the CM tool commands, no need for knowledge of the CM structure, an easy access for users outside of Siemens ICN ICM CA R, and the platform/ editor independence realization are the high level goals reached by this tool.
The WATCH main features include:

- *Easy Query* that enables the document retrieval from CM environment with an easy web interface;
- An *Advanced Search* engine, in accordance with advanced selection criteria, lists the documents with indication of matching likelihood;
- *Code Patterns matching* on source code for occurrences of variables, function, comments etc… makes WATCH a powerful tools as support to software reuse activities.

The work-flow foresees that a user, looking for existing documents satisfying simple, pre-defined criteria, receives in output a list of matching documents. Once the user has found out the document he/ she is interested to, selection happen by clicking on Open or Download button. For each displayed document the related Product, release, Item type and the title are provided.

Statistics reports, tracing the access made with respect to product, release, document type, user group action performed and type of search provide useful information on DB and Network loads.

As an example in the Fig. FB.6 the result of a basic search is shown. Referring to the Project BSC-LMT-TRAU, Release X.Y, WATCH shows that four items are available under the configuration management repository and classified as Feature Sheet Document.



Project: BSC-LMT-TRAU

Release: BR2.0 BR2.1 BR3.0

Item type: FSH - Feature Sheet Document

Description	Item ID	Release	Item type	Rev	Status	Action
Item F_BSCFH_0261.255	F_BSCFH_0261	BR5.5	Feature Sheet Document	5	APPROVED	
Enhanced MicroBTS	F_BSCFH_0363	BR5.5	Feature Sheet Document	2	APPROVED	
Item F_BSCFH_0720.255	F_BSCFH_0720	BR5.5	Feature Sheet Document	5	APPROVED	
GPRS: Performance Measurement	F_BSCFH_1868	BR5.5	Feature Sheet Document	5	APPROVED	

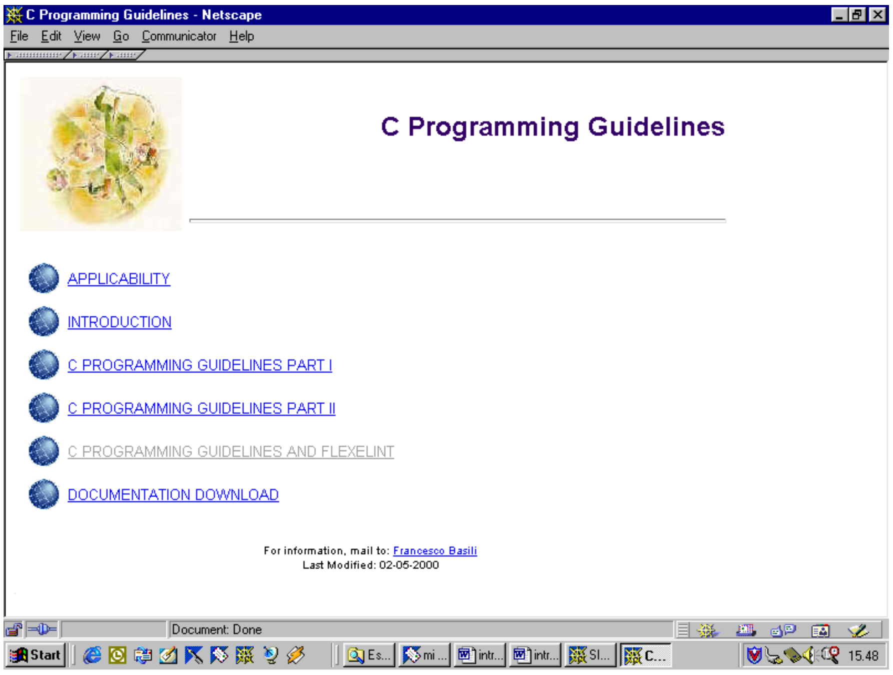
[1] 4 items found Page 1 of 1

Fig. FB.6 WATCH basic search result

On line training

A repository of all the training courses for new Siemens employees documentation is quickly accessible from the Intranet. Four main areas of interest have been singled out for training: the quality area, the development process area, the system level area and the technical area. For each area a set of training courses have been established.

Just as an example report the C Programming guidelines web section that contains all the company rules and recommendations, established at project level, that shall be followed by the programmers during the coding phase.


C Programming Guidelines - Netscape
File Edit View Go Communicator Help
Document: Done
Start
15.48

C Programming Guidelines

APPLICABILITY

INTRODUCTION

C PROGRAMMING GUIDELINES PART I

C PROGRAMMING GUIDELINES PART II

C PROGRAMMING GUIDELINES AND FLEXELINT

DOCUMENTATION DOWNLOAD

For information, mail to: Francesco Basili
Last Modified: 02-05-2000

Fig. FB.7 The C Programming guidelines intranet page

The programmer, using this services, is able to easy retrieve the established rule for example the naming convention, file structuring, project specific definitions, code comments, file and function header fill templates, simply using its web browser. The users are also able to download the file and function header templates for an easy cut and paste within their own code.

Lessons learnt and future goals

Intranet services are very successful and their usage has become commonplace and popular within the technical staff, it should be clear that the service development requires specific skills and know-how, so it is important to keep a distinction between info providers and web-master. Intranet is becoming the unifying element among distributed sites and heterogeneous work environments.
Some of the lessons learnt are:

- the importance of active usage of mailing list and of their updating;
- document repository instead of mails with attachment;
- usage of both NT and Unix servers: each one for their best features;
- take care of impacts on network infrastructure;
- importance of visibility levels and user authentication;
- forms require work-flow management;
- trade-off between central and bottom-up initiatives;
- integration with corporate wide intranet portals.

In the future the Intranet evolution will occur in planned steps; further improvement of all available services is foreseen other than the provision of additional workflow management and the realisation of a powerful search engine.

Appendix A: authors short CV

Francesco Basili (fbasili@onion.it) received his degree as Electronic Engineer at University of L'Aquila in 1997. He has been working for ONION S.p.A. since 1998 in the consulting department focusing on Software Process Improvement activities. In this context he co-operates with customers (especially in the telecommunication domain) in the fields of software testing and quality assurance . His research interests include software process engineering, test automation, software reverse engineering and software reuse.

Gualtiero Bazzana (gb@onion.it): born in 1966, at university, he graduated with 110/110 and honour in Information Science at the University of Milan, in February 1989. His PhD won the special AICA award for topics related to quality in Information Technology. After working as software developer in a telecommunication company and as consultant/ manager in a consulting company he set-up ONION, where he is Partner/ Member of the Board and Business Management Director.
His activities cover two areas of interest: consulting - projects in software engineering for various industrial companies and research in the field of software quality and networking (especially in the Internet/ Intranet domain). He has co-ordinated several European Research Projects. Moreover, he has published a book: ("Software Metrics for Product Assessment", McGraw Hill, London, 1995, International Software Quality Assurance Series, ISBN 0-07-707923-X), contributed to 4 other books (in the last one he has been author of four chapters dedicated to Software Process Improvement; it has been published by IEEE Software) and published over 50 papers at international conferences on topics related to software quality and software testing.

Giannino Zontini (gz@onion.it) is 32 year old IT professional with 9 years of programming experience. Developed various packaged products in Oracle environment with particular reference to the ERP. Now specialising on Microsoft web technologies - ASP, DHTML, VBScript, JScript, XML, IIS, SQL Server etc.
In recent years he's specialized in designing Internet and Intranet systems developing Web based applications and interfacing databases to the Web primarily using Active Server Pages technology. Currently working as a software consultant for Onion S.p.A focusing on providing E-Commerce Solutions through the integration of SAP R/3 to Internet.

Roberto Delmiglio (roberto.delmiglio@icn.siemens.it)
Sergio Scotto Di Vettimo (Sergio.ScottoDiVettimo@icn.siemens.it)

Appendix B: short companies profile

Siemens Information and Communication Networks SpA is a subsidiary of Siemens AG of Germany, a $74 billion (FY 98-99) global organization with 440,000 employees in 193 countries.
Siemens is a technical innovator, one out of every nine employees is assigned to research and development, a quality manufacturer of electronic products, from communications and information technology to healthcare, transportation, energy and power. Siemens Information and Communication Networks SpA designs, manufactures, markets and installs systems and equipment for public and private applications. In Italy and abroad it implements systems and networks on a turnkey basis.
Siemens Information and Communication Networks SpA is structured in five business areas: *mobile networks, radio networks, optical fiber networks, multi-service networks and IP and Corporate Networks.* The company is active, full-line, in every field of telecommunications.
The Mobile Networks Business Unit has the following mission:

- to be a turn key mobile networks supplier;
- to develop the business of mobile networks;
- to be the competence centre for the whole Siemens group for Base Station Subsystems used

within cellular networks and wireless access systems;
- to support customers in all project implementation phases.

ONION S.p.A., founded in 1994, is a privately owned company based in Italy offering advanced services in the Information Technology domain; the company is specialised in the field of ICT, SW technologies, consulting and ERP systems, always trying to keep a strong link between technology and business goals.
ONION S.p.A. is the head company of the ONION Group (www.oniongroup.com) that also includes:

- Proxima Ventura, specialised in E-business solutions
- Vision Partners, specialised in Help Desk/ Call Centre Solutions
- Marilland, acting as commercial focal point for the whole group

The goal of ONION Group is to be an excellence centre for high technology applied to business processes. In this context ONION strategy is to act as technology provider, system integrator and consulting provider for medium and large companies in Italy and abroad.

References

[1] *"Intranets: the new engine for Corporate Productivity"*, white paper - David E. Welch

All the other information have been extracted from technical documentation provided by Siemens Information and Communication Networks S.p.A

The Victory Project - A Virtual Enterprise for SPI

Dr Richard Messnarz
ISCN LTD, Ireland
Dr Miklos Biro
Sztaki, Hungary
Gonzalo Velasco
General Fundacion, Spain
Gearoid O'Suilleabhain
DEIS, Ireland

Introduction

The paper represents a first dissemination of the Victory initiative in which projects who developed concepts for learning organisations, and who offer methodologies for process improvement contribute to the establishment of an SPI (Software Process Improvement) E-Commerce site to make solutions available and usable across Europe via the net.
The first content will be based on two previous Leonardo da Vinci initiatives (PICO and Bestregit) who developed business strategies for SPI as well as a concept of a constantly improving learning organisation. It will also support the creation of a body-of-knowledge for the EuroSPI initiative by making SPI experiences across Europe accessible with search and rating services.
In future the content will be opened to further initiatives to establish a virtual European SPI body of knowledge.
The project started in December 1999 and a first version will be online in Jan. 2001.

SPI Knowledge

What means SPI Knowledge ?

If you perform SPI (Software Process Improvement) this mostly relates to changes of the organisation, processes, skills, and infrastructure with the goal to increase the reliability and cost efficiency of a firm , to reach a larger business share, and to reach a higher customer satisfaction.
Once an SPI project is successful it leads to knowledge (commonly described in terms of processes) which shall be shared amongst many projects. Knowledge in this respect, however, is much more complex than just the model of a process. It includes factors such as

- Strategies (market objectives, strategic decisions)
- Goals (aligned to business objectives)
- People (skills and technical competence, motivation, personality, roles)
- Processes (models, environments, etc.)
- Measurement
- Interdependencies (with other groups)
- Etc.

It happens that if you loose out one factor you might loose with SPI.

e.g. concerning strategies

If the strategy is wrong the whole improvement is a waste. In [PICO 1999] it is described:

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

So if SPI would just try to further increase the maturity level of this European manufacturer and introduce even more processes they would fail and money would be wasted. But if they learn to tailor their standards to different quality levels of products they can compete.

e.g. concerning goals

If the strategy is right, and the goals are wrong the whole improvement is a waste. In [PICO 1999] it is described:

A big electronics company in Europe, for instance, made an assessment resulting in maturity levels for different areas of the organization and the identification of weaknesses such as unrealistic planning, no process for design reviews, and weak configuration management. The organization was already ISO 9001 certified but only 30% actually accepted the guidelines due to missing practicability (formally good documented but not realistic for projects in the field). A formal pragmatic assessment and improvement approach would then, for example, decide about introduction of configuration management and so forth, BUT does this really now meet the organization's business goals? Without

any additional data than the normal assessment methodologies (CMM, Bootstrap, etc.) this unfortunately cannot be answered ?
So this electronics company decided to run a goal analysis (based on the GQM approach) in parallel interviewing business managers, department heads, IT managers and project managers and designing a consistent goal tree from top to bottom.
One of the specific business policies was to create a financial framework for the next years which allows to get a reserve budget to fight for a brand new product on the market. To get to this marketing budget the business managers decided to stabilize the development effort from divisions at x% so that with all other overheads and cost a certain percentage is saved every year to have the budget together right at the time when the product is announced. At this moment the divisions were certainly higher than x% and the improvement actions (based on the previously identified weaknesses from the assessment) had to demonstrate after 3 years the success of achieving this business goal.
The technical staff were frightened and thought that people will be dismissed but the truth was that a proper interpretation of the business goals led to a completely different view. The business managers expected that process improvement provides a better work process and environment so that with the same staff more projects and tasks can be done and over time the development effort is stabilized at x%.
Under this perspective the 3 process weaknesses were again analyzed and further interviews showed a potential of re-use because in all systems in the sector nearly 80% of the functionality was always the same. So the improvement plan focused on an integration of design, configuration management and review of a re-use pool of these 80% functions and to reduce the development for each project to the only 20% additions, thus enhancing productivity and achieving the effort stabilization.
Now, let us assume that only a pragmatic assessment would have been performed. Three weaknesses would have been identified and without re-use orientation would have led to a pragmatic proposal to first run a pilot project to identify a configuration management system and field test it, to disseminate it to other projects, and to help making it a division wide standard. Sounds simple, BUT unfortunately the business context is lost. What then happens is that management sees additional effort, the development effort further increases, and with no vision of decrease of the development effort the business manager after 1 year (before benefits can be made visible) would really decide about things like dismissals.

So if SPI would just try to follow each time the same pragmatic steps without taking into account the specific business environment and goals it will highly likely fail.

e.g. concerning people

Even if you have the best processes they can fail due to many reasons [Bestregit 99, Bestregit 2000, Hofstede 80, Osuille 2000, Siakas 95]
- Psychologic reasons: wrong leadership style, no motivation, defense mechanisms hindering change
- Skill and personality reasons: even with the same background people have varying productivity.
- Cultural aspects: misunderstandings, language problems, different work styles

So knowledge and competence can be lost even if defined processes are in place and people create problems or leave the firm. In this respect the Bestregit project worked on team-work and learning concepts and [Bestregit 1999, 2000] found out that if the proper communication and team-work scenarios are in place the time to integrate new staff members can be reduced to one third of the previous time. However, this requires team-work based processes [Messnarz 99] drawing on roles and communications instead of just listing a number of process steps (as still many organisations do). A team-work view is important that people feel integrated, satisfied and understand how they can contribute to the team's success.

Also too many organisations following the pragmatic SPI approaches [MacAnAirchinnigh 1996] forget that most innovations start in the heads of single super-intelligent people, who then need much time to convince a critical mass of people, and only after this critical mass is reached the company starts to support this new line. And these innovations later form major business lines of the firm.
Even if organisations then efficiently follow a people CMM approach [Ericsson 99, Curtis 95] and are not aware of the value of such extra-guys (who normally do not fit in the scheme) they will loose these

innovations to the competitors.

An interesting example is that in EuroSPI 1999 there were presentations from Ericsson and Nokia. Ericsson (years ago) had a majority of the market shares on the mobile market (above 50%) and started with pragmatic CMM, and Nokia invested much in the cooperation with innovative groups in Scandinavia pushing the innovation of their products. Meanwhile Nokia just was at CMM level 1 and took large market shares while Ericsson working on standardisation and CMM schemes lost market shares. Of course, there are many outside factors that can influence such a situation, but still it sounds interesting.

e.g. concerning interdependencies

Although not visible for the customers (e.g. cars) there are complex supply chains. A car manufacturer outsources the software development to a supplier who in return works with further software sharing partners and suppliers.
In the Bestregit project [Bestregit 1999, Bestregit 2000] we field tested how process models and plans work in interdependent services and development. We started from the ideal model drawn and measured duration times and effort, comparing the expected duration times and effort with the actual ones.
The times which were planned for interfaces with partners needed double the time expected, while all internal steps were nearly accurately estimated. And these times were due to waiting times for supplier deliveries, increasing the total time by 17% for the project. So it means that *ideal process models* in supplier chains will fail if the processes are not calibrated to realistic buffer times, effort and duration.

SPI Knowledge Definition

Thus SPI knowledge is not just
- To use a CMM methodology [Paulk 93]
- To follow an ISO or ESA or other standard [ISO 9001, 9000-3, 9126, 12207..]
- To use the Bootstrap methodology [Messnarz 94,95,96]
- To use a GQM approach [Debou 94, Basili 98]
- Etc.

It is rather the knowledge of how the different factors influence each other in certain industrial cases which can be applied for your own organisation. This requires that an inter-disciplinary, inter-methodology access to know how is enabled that lets you search a big set of experience cases and analyse
- Which combination of existing methodologies can be used in your specific situation [PICO 99]
- How the strategy relates to goals
- How the goal relates to processes
- How the processes relate to team-work and people factors
- How the interdependencies influence the cooperation success

This then requires
- A large set of experiences
- A guidance to browse this experience
- A support to select the right experience which can work for you
- A guidance to combined usage of methodologies and approaches as required to satisfy all the factors (as discussed above)
- An access to the different methodologies
- A ready to use platform to offer SPI products and knowledge in an e-commerce environment

An SPI Body of Knowledge

This section describes an Internet based Process Improvement E-Commerce initiative Victory, and it also describes which long term goals are planned in cooperation with the EuroSPI consortium.

The Victory Leonardo da Vinci Initiative

Victory is a multiplier project for two previous Leonardo initiatives, PICO and Bestregit.

PICO stands for Process Improvement Combined apprOach. What makes PICO special is (in comparison to other Software Process Improvement initiatives) that [PICO 99, 2000]

- PICO established an integrated approach as a combined use of the existing methodologies on the market focussing towards business goals and achievements. Many business cases and business figures are reported in a book and the courses.
- PICO established a tool set which can be adapted to different assessment methodologies and evaluation algorithms by configuration. So PICO is also a platform technology provider for most of the existing assessment methodologies.

This project involved 30 experts from 11 EU countries who produced:

1. Marketing and broad dissemination of a **book**. Available through an electronic book store on the WWW managed by the IEEE Computer Society Press.
2. **Six developed one day courses** which focussed on different business and success factors which are important in the implementation of software process improvement. The courses are available with trainer notes, students notes, and slides. The trainer notes allow that trainers can receive the course and hold it themselves for their staff or clients.
3. A **tool set** configured with the most common assessment methodologies, such as ISO 9001, CMM, Bootstrap, SPICE, and any others (if available).

Target Group:

The target group are division managers, project managers, quality managers, auditors, and consultants in the IT sector.

Bestregit stands for BEST REGIonal Technology Transfer. Bestregit developed a simple to follow approach for innovation transfer organizations to improve their innovation services and achieve the structure of a continuous learning organization. Later in the project it was found that the approach is beneficial for service organizations in general and has been field tested with service (government owned and private) organizations as well. [Bestregit 99, 2000]

Results which can be multiplied and disseminated:

1. An **improvement guideline** with clear guidance, steps, and examples from the user organizations and field tests, to establish an improved organizational structure with a business focus and a learning organization culture. The difference to PICO (and the usual assessment methodologies) is that Bestregit much more focuses on team and people aspects, and speaks a language which can be understood by non IT people. It is a framework that was successfully field tested with people with no IT background like sociology, business, or languages, or politics.
2. A set of **case studies** from the user organizations who field tested the entire guide-line and its steps and helped to refine it. We might use the measurements done in these studies for show cases and commercial argumentation.
3. An **interactive course** (which got very positive feedback in the field tests) which demonstrates the improvement steps, the learning organization model, and supports the participants in the application based on real examples (their own organizations).

Target Group:

The target group are directors from SMEs and service organizations, innovation managers, managers in government agencies, and managers in general.

Victory's goal is to

- Establish a commercialization plan and strategy for the PICO and Bestregit results
- Design a WWW and E-Commerce site for a Europe wide distribution
- Translate the products into German and Spanish
- Evaluate the market impact and sustain the E-Commerce approach for further EU Leonardo results (a platform for EU results in the Leonardo da Vinci Programme and for industrial experience in SPI)

Allied Initiatives

EuroSPI is a consortium of large Scandinavian research centres, a large German quality assurance association, the software engineering centre of the MOD in UK, and ISCN as the coordinator of the team. EuroSPI bases on a conference initiative which started in May 1994 and has since gathered hundreds of well described experience reports. Also each partner of EuroSPI was involved in additional SPI initiatives comprising further some hundred experience reports. E.g. ISCn was partner in EPIC where we gathered about 150 industry reports.
This experience base will be integrated into the SPI body of knowledge and made accessible with query and rating mechanisms. A first version is already at
www.iscn.com/select_newspaper/select_index.html

Select is a European initiative of partners of the W4Group consortium who develop new Internet protocols and services for rating and guided Web browsing. This group supports the creation of this body of knowledge by combining this large set of experience reports with rating and search services.

A First Architecture

The architecture is designed in such a way that the previously mentioned requirements can be satisfied and the scope of the Victory plans is met:

SPI Knowledge requires
- *A large set of experiences*
- *A guidance to browse this experience and a support to select the right experience which can work for you*
- *A guidance to combined usage of methodologies and approaches as required to satisfy all the factors (as discussed above)*
- *An access to the different methodologies*
- *A ready to use platform to offer SPI products and knowledge in an e-commerce environment*

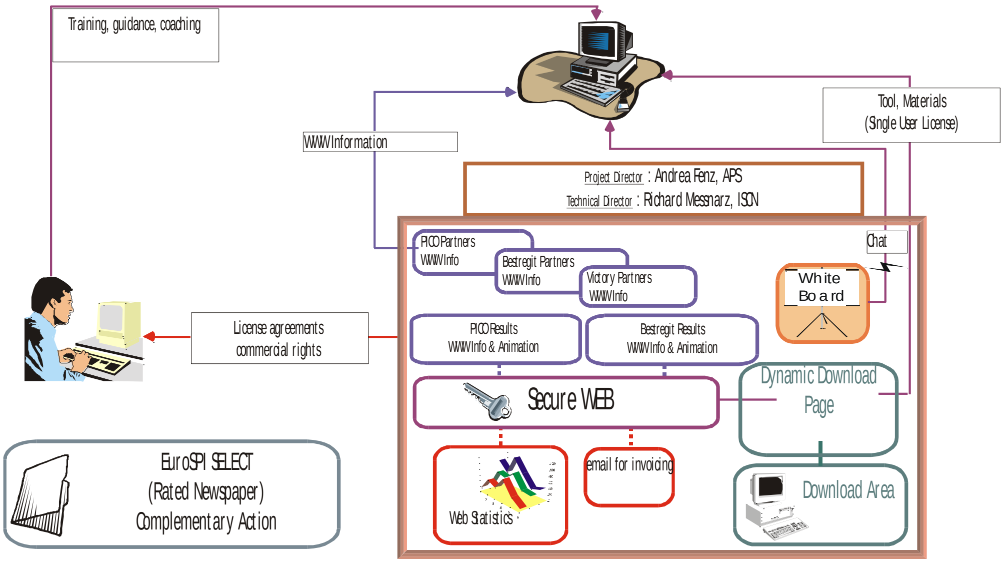


Figure 1 : Victory E-Commerce Architecture

A large set of experiences is represented by
- The PICO book (can be ordered online from the IEEE publisher) written by 30 experts from 11 countries of the EU with contributions from leading European companies
- The EuroSPI proceedings are going online
- Previous ESSI initiatives (e.g. EPIC) will be linked and integrated

This will help to achieve a library of some hundred well elaboarted and reviewed experience reports.

A guidance to browse this experience is elaborated by the project SELECT as well as by the Victory partnership applying animated design tools.

For the selection of the right set of methodologies and experiences we follow a two-fold strategy:
1. PICO offers guidance about how to select the right methodology based on your business case (from a methodology viewpoint).
2. The Victory site will offer a search mechanism to select a set of experience reports which seem applicable for your business environment.

All relevant method providers will be linked and experience reports will be associated to them. We also will offer chat rooms on certain SPI topics and experts available at certain times.

And most important, to offer a platform concept with ready to use e-commerce components which allows EU projects and providers to insert their services and products into a e-commerce environment offering
- information services
- ordering services
- customer database services
- dynamic baskets (compare amazon) to allow people shopping of SPI solutions on the net

This platform concept will be developed in such a way that it can easily be enhanced to further initiatives that want to be included.

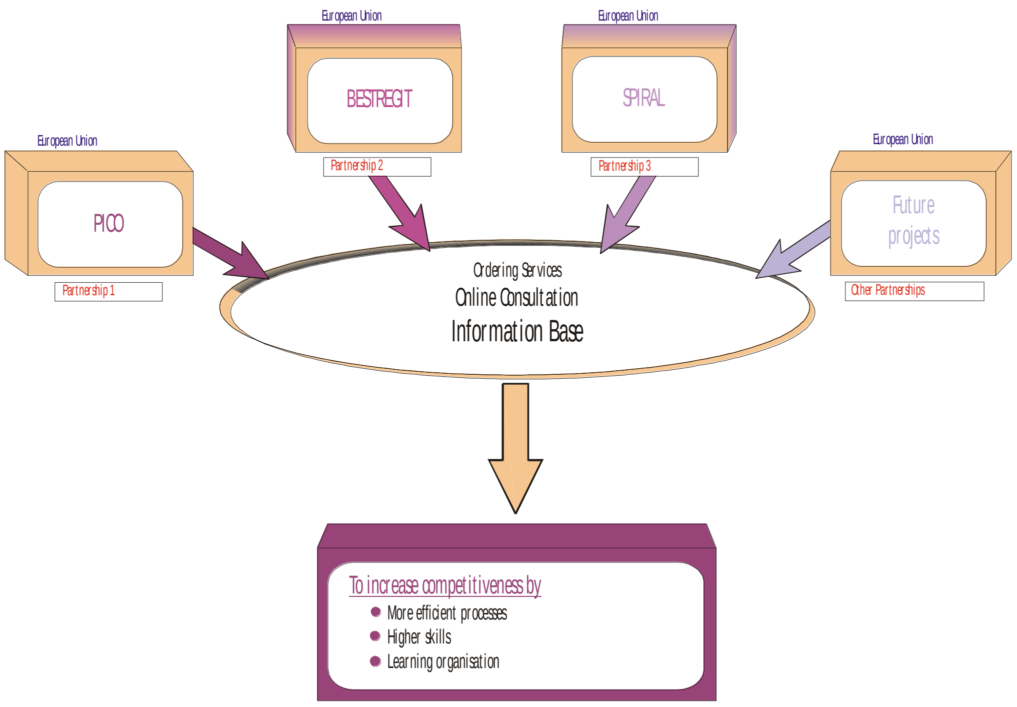


Figure 2 : Victory - A Platform Open for Further Initiatives

A Long Term General Concept

Once Victory runs for the Leonardo initiatives in SPI, we plan to enhance this body of SPI knowledge to involve all EuroSPI partners and their services, products and initiatives. As EuroSPI enhances by 2 countries (funding partners) per year, and as EuroSPI is already funded until 2003 and planned until 2010, this gives a long term background and the chance of covering all European countries with a regional representative per country (as the regional excellence centre).

If this works out this could have an impact on the EU beyond 2010 because at this stage Victory can start collecting all regional experiences and summing them into one European body of SPI knowledge.

The advantage of such a bottom up approach is that it naturally grows (like a flower) and is a step by step integration instead of a big hammer top down all-at-once initiative.

In the Bestregit project [Bestregit 1999, 2000] a learning spiral for SPI was identified which usually organizations run through when applying a combined set of SPI methodologies, and the learning phases were aligned with leverages (business levers) a business manager wants to achieve.

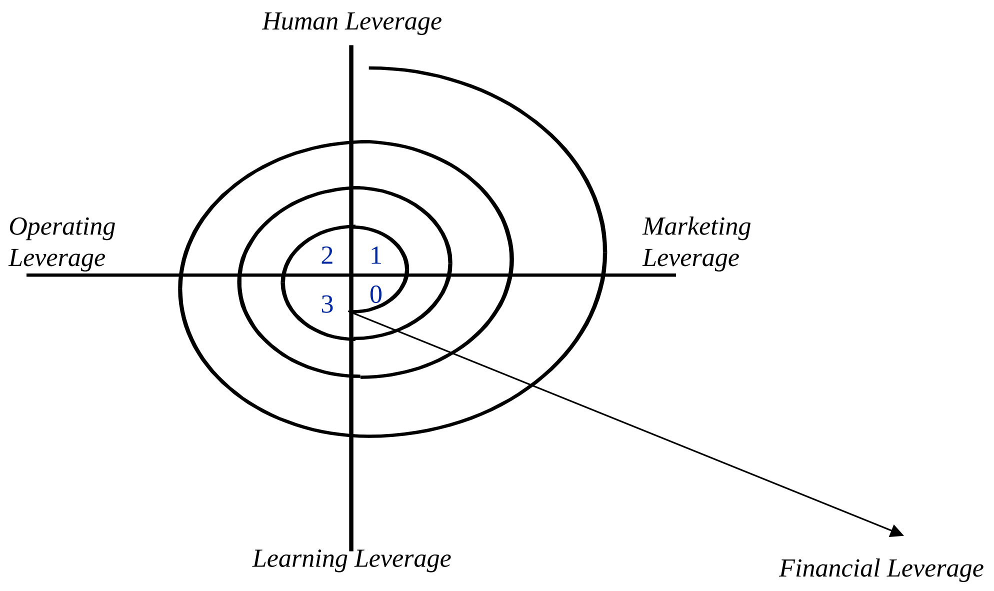


Figure 3 : The Learning Spiral and Business Levers

Why do managers invest into improved skills and processes ? [CapersJ 96, Herbsleb 94, Bestregit 99]

Operating leverage is related to the cost structure, that is the repartition between the fixed costs and variable costs. Process improvement means an increase in fixed costs (training, consulting fees, equipment, improvements in office conditions, etc...) to decrease the variable cost (average effort to do a certain service at a give quality).
If, due to process improvement, the firm is able to deliver the same quantity and better quality of service using less person months than earlier, then it will have the potential to take advantage of operating leverage.
Learning Leverage - It is an empirical fact that unit costs decline exponentially when experiences are accumulated and the steady reuse of these experiences is well managed by the firm. This is called production leverage in the manufacturing industry, while learning leverage is a better expression in the service sector.
Marketing Leverage - Process improvement, maturity achievement, ISO 9000 certification have an important impact on the perceived capability of the company and on the perceived value of its products or services, which contributes to improved customer satisfaction.
Human Leverage - It is widely known that employee motivation (empowerment) can be significantly influenced by immaterial means like management styles and organizational structures. This means that the same employees can perform more work and even in better quality than otherwise.
Financial leverage means borrowing funds and investing them with a return higher than the cost of the debt. If a company is able to exploit financial leverage, it can make money on funds it does not own.

If organizations apply a combined set of methodologies as described in the PICO materials (for software organizations) and in the Bestregit materials (for service organizations) they usually

- kick off an SPI initiative increasing the customer's trust and their image leading to a higher marketing leverage (step 0 in Figure 3)
- Analyse the goals and align strategy with business and work goals achieving a common mission for the teams, leading to a higher human leverage (step 1 in Figure 3)
- Establish processes (supporting team cooperation) to work more efficiently aiming at operating leverage (step 2 in Figure 3)
- Experiment and measure if the processes have to be calibrated or changed (step 3 in Figure 3) leading to a learning leverage

The overall goal is to achieve a financial leverage as a third dimension for the business managers.

If the SPI body of knowledge offers a set of different methodologies and provides guidance about how to establish a combined set serving your goals best this learning spiral of firms should be represented in the learning model offered to the users.

These users then connect the general Victory site (with experience from different partnerships, sources, countries) and use the experience library, the method and tool sets offered, together with the guidance o follow his specific learning spiral.

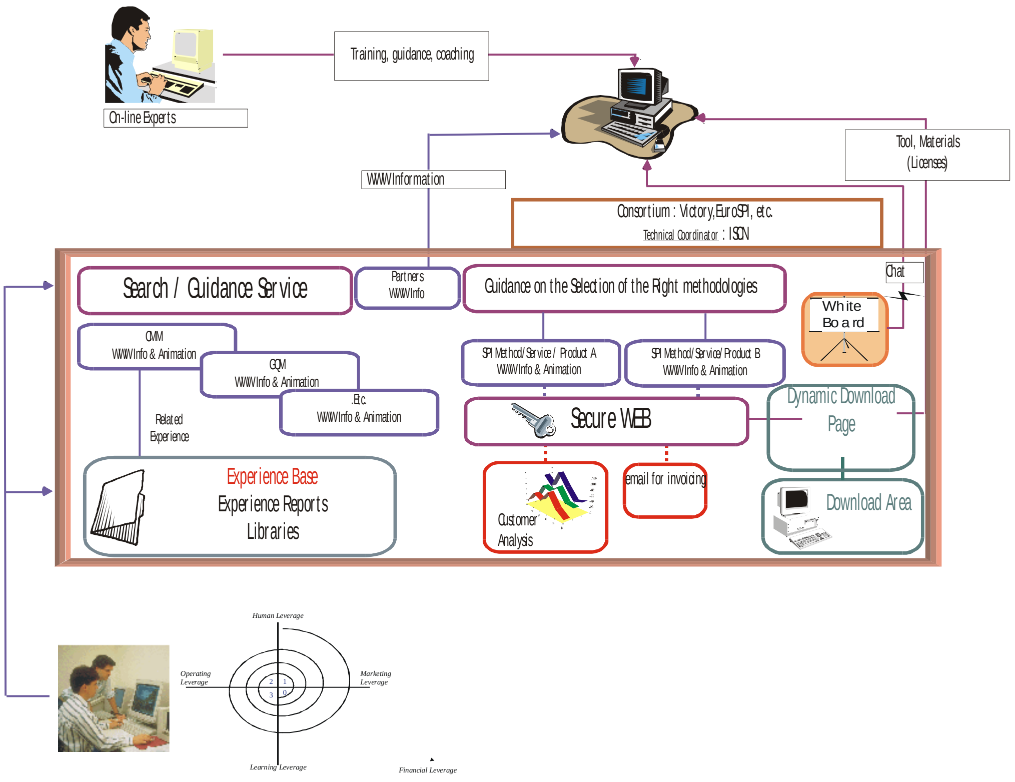


Figure 4 : A General Victory Architecture

As Leonardo focuses on a Long Life Learning Europe the initiative will support a long life learning process for SMEs across Europe in the SPI sector.

Acknowledgements

We are grateful to the work of all experts who contributed to the PICO initiative from 1995 to now. We are grateful to the work of all experts who contributed to the Bestregit initiative from 1997 to now.
We are grateful to the Leonardo da Vinci Programme of the European Union who helped us to create such a critical mass of development in the SPI area, and for supporting Victory.
We are grateful to all EuroSPI partners, and to ESI and SP96 partners from previous ISCN conferences, who helped us to collect so much relevant experience over the last 7 years.

References

[Basili 98] Basili, V., Rombach, H.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, Vol 14, Number 6, June 1988.

[Bestregit 99] A Learning Organisation Approach for Process Improvement in the Service Sector , R. Messnarz. C. Stöckler, G. Velasco, G. O'Suilleabhain, A Learning Organisation Approach for Process Improvement in the Service Sector

[Bestregit 2000] The Bestregit Improvement Guideline - Learning to Improve, Bestregit Final Report, CEC-Leonardo da Vinci Programme, April 2000

[CapersJ 96] Capers Jones, The Pragmatics of Software Process Improvement. Software Process Newsletter. No.5, Winter 1996, pp.1-4.

[Curtis 95] Curtis Bill, Heflleey William E, Miller Sally. Overview of the People Capability Maturity Model, CMU/SEI_95-MM-01, 1995

[Debou 94] Debou C.: ami a new paradigm for software process improvement, In: *Proceedings of the first ISCN Seminar*, Dublin, May 1994

[Ericsson 99] G.Evangelisti, E. Peciola, C.Zotti, People Management and Development Process to motivate, develop and retain the best resources, in Proceedings of the EuroSPI 99 conference, 25-27 October 1999, Finland, also published at the EuroSPI web site.

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

[Hofstede 80] Geert Hofstede, Motivation, Leadership, and Organization: Do American Theories Apply Abroad? Organizational Dynamics. Summer 1980, pp.42-63.

[ISO 9000-3] 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).

[ISO 9001] ISO 9001. Quality Systems. Model for Quality Assurance in Design/Development, Production, Installation and Servicing. International Organisation for Standardisation, Geneva (1987)

[ISO 9126] ISO 9126, Information Technology - *Software Product Evaluation* - Quality Characteristics and Guidelines for Their Use, 1991

[ISO 12207] ISO/IEC 12207, *Information technology - Software life cycle processes*, first edition Aug. 95.

[MacAnAirchinnigh 96] M. M. Airchinnigh, The Place of Formal Methods in Process Improvement, in ISCN'96/SP'96 Proceedings, 3-5 December 1996

[Messnarz 94] 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

[Messnarz 96] Messnarz R., Practical Experience with the Establishment of Improvement Plans, in: *Proceedings of the ISCN´96/SP'96 Conference on Practical Improvement of Software Processes and Products*, ISCN LTD, Brighton, UK, 1996

[Messnarz 95] Messnarz R., Scherzer H., The Evolution of a Quantitative Process Analysis - the BOOTSTRAP - Approach, in: (eds.) G. Chroust, P. Doucek, Interdisciplinary Informational Management Talks, Oldenbourg, Vienna, Munich, 1995

[Messnarz 99] Messnarz R., Stubenrauch R., Melcher M., Bernhard R., Network based teamwork and Quality Assurance (NQA), in: Proceedings of the 6th European Conference on Software Quality, Vienna, April 1999

[Osuille 2000] Gearóid Ó Súilleabháin, MScEcon., DEIS, Cork Insitute of Technology, OSIRIS Project - Cross-cultural Collaboration: the experiences of the LEONARDO ISIS project, May 2000

[Paulk 93] Paulk M. C., Curtis B., Chrissis M. B.: Capability Maturity Model for Software, version 1.1, CMU/SEI-93-TR-24, February 1993.

[Paulk 93] Paulk M. C., Weber C. V., Garcia S. M., Chrissis M : Key practices of the Capability Maturity Model, version 1.1, CMU/SEI-93-TR-25, February 1993.

[PICO 99] Better Software Practice for Business Benefit - Principles and Experience, eds. R. Messnarz. C. Tully, IEEE Computer Society Press, Washington-Tokyo-Brussels, November 1999

[PICO 2000] M. Biro, R. Messnarz, Key Success Factors for Business Based Improvement, in Software Quality Professional, Volume Two, Issue two, American Society for Quality, March 2000

[Siakas 95] Mohamed Walaa Eldeen and Siakas Kerstin V. (1995): Assessing Software Quality Management Maturity (SQMM): A new model incorporating technical as well as cultural factors, the 3rd International Conference on Software Quality Management SQM95, 3-5 April, 95, Seville, pp. 325-336

Session 5 - SPI and Personal Processes

Session Chair:

Richard Messnarz,

ISCN

Improving Individual Software Engineering Skills

Maurizio Morisio, David Escala
Politecnico di Torino, Italy
Gerry Coleman
Dundalk Institute of Technology, Ireland
Marisa Escalante, Cliona McGowan
ESI, Spain

Christophe Mercier
AFTI, France
Rory O'Connor, Howard Duncan
Dublin City University, Ireland
Yingxu Wang
IVF, Sweden

Introduction

The software crisis is not over. Glamorous failures, such as the Ariane 5 or Mars Orbiter, happen regularly. But also "normal" projects are affected. Each project manager has experienced, directly or indirectly, budget and schedule overruns in software projects. A recent study in the UK [Taylor 2000] unveils that only 13% of IT projects are successful. Of the IT projects classed as successful less than 1% were software development projects.

Technology can help to tackle the problems, but, ultimately, people and organisations produce software. This straightforward observation has led to focus improvement on the software process. From this point of view, we can recognize two intertwined trends:

- Software process improvement at the organisation level (such as the CMM, the ISO9000 series, Spice (now ISO15504).
- Software process improvement at the level of individual and small teams.

Process improvement at the organisation level suffers from some drawbacks. It is usually a very top-down approach, that gives little support to individuals and risks not addressing their day to day problems; and it is cumbersome and especially ill-adapted to SMEs.
For these reasons other approaches have been defined to support individuals and small teams. As a result these approaches are much more suitable to SMEs. Watts Humphrey has initiated this trend

proposing the PSP [Humphrey 1995] and the TSP [Humphrey 1999]. Kent Beck proposes Extreme Programming [Beck]. And our consortium proposes PIPSI, an approach which will be described later.

The PSP legacy

PIPSI is greatly inspired by the PSP, so in this section we review what the PSP is, and how PIPSI derives from it.
The PSP was developed by Watts Humphrey to address the deficiencies of the CMM at the individual level. It is based on a set of key concepts that are aimed at the individual developer.

- The process: each developer should follow a defined process
- The measures: each developer should monitor his or her performance by using a set of measures. The same developer collects the data and analyses it.
- Estimation and planning. Each developer should use adapted techniques to estimate the duration of his project, plan it, monitor its advancement and compare with the plan.
- Quality. Each developer should use specific techniques to improve the quality of her project. The goal is zero defect software. Defects should be eliminated as soon as possible, possibly before the first compile.

The PSP includes a model to teach it and introduce it in use. Teaching is based on a two weeks course, which includes completing seven to ten small projects. Practice is key to adopt the habit of applying the PSP, but usually lengthens the duration of training to close to three weeks. After the course the developer goes back to her company and is expected to start using the PSP. Although the PSP is personal, Humphrey recognizes the importance of backing from management to foster its day to day use.

The results from applying the PSP are contradictory. In classroom setting [Hayes and Over 1997] developers reduce the number of defects they leave in their programs while not changing their productivity. In industrial setting [Humphrey 1996, Ferguson 1997] estimation accuracy improves, and the same happens to the quality of the product.

However some problems are reported too. El Emam [El Emam 1996] reports a high rate of individuals who abandon using PSP having attended PSP training and subsequently been unable to apply it in their workplace. Morisio [Morisio 2000] reports that the duration of training is often unsustainable for SMEs, tool support is essential, human factors (such as the privacy of data) are key, and finally that the PSP cannot be applied to individuals in isolation when they work in cohesive groups.

Based on findings from literature and on the direct experience of the IPSSI partners, it was clear that the PSP needed a number of modifications to become more suitable to industrial applications, especially in European applications. These modifications included:

- Reduce the duration of training, to make it more affordable
- In general, simplify the contents of the PSP, although keeping the original concepts
- Provide tool support for collection of data, computation and analysis of measures
- Address human factors, build data privacy in the supporting tools
- Don’t overlook the transition phase from training to day to day usage in industry

The IPSSI project

The IPSSI (Improving Professional Software Skills in Europe) project is an EU funded project which provides a process improvement framework for use by individual software engineers working in European SMEs. The focus of the project is on improving individual software engineering skills thus generating bottom-up improvement. Companies can experiment with SPI by sending individuals on

IPSSI training courses and then monitoring its implementation. By training individuals in this way, costs are reduced and the quality of products improves.

IPSSI has produced two main deliverables: a set of training materials (PIPSI) and a supporting tool (TIPSI). This paper is dedicated to PIPSI, a companion paper discusses the tool.

Project leader for IPSSI is Dublin City University. The other partners are AFTI (Paris, France), the European Software Institute (Bilbao, Spain), Politecnico di Torino (Torino, Italy) and IVF (Gotheborg, Sweden).

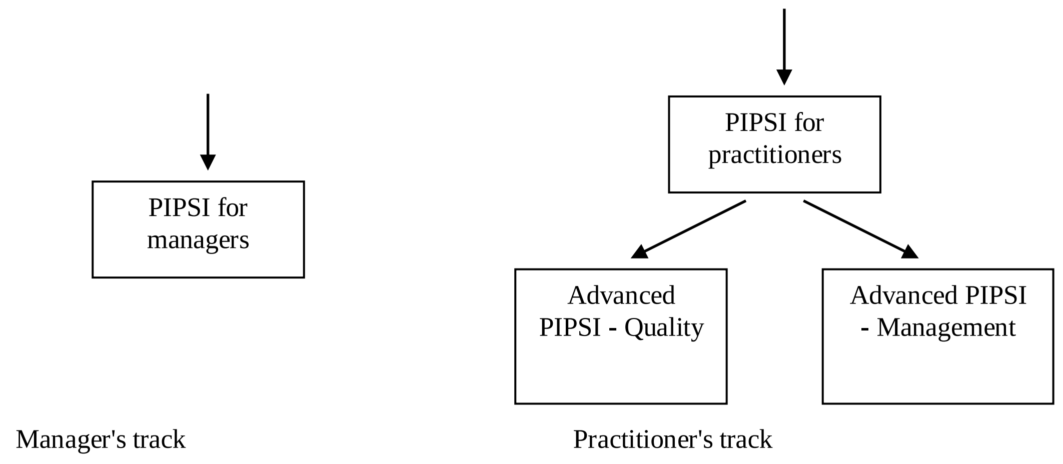
Pipsi

The introduction path

The final goal of PIPSI is to be used in day to day work, to improve it. To achieve this goal, an introduction path has been defined. It consists of three phases

1. Motivation
2. Training
3. Transition to day to day use

The first two phases are supported by training materials and the TIPSI tool. The latter phase is supported by consultancy. TIPSI is available as open source, it can be used as is in phase 3, or parameterized, or modified.



Motivation is supported by the course PIPSI for managers (one day). The course, targeted to decision makers of companies, describes the concepts behind PIPSI and the rationale for using it. Training is supported by three courses. PIPSI for practitioners (two days), targeted to developers, teaches the main techniques and concepts. It is a hands-on class, where developers apply what they learn on five short programming exercises. As a result, developers build a baseline of personal measures too. Finally, two Advanced courses (one day each) teach advanced techniques for personal quality management and personal project management. The Advanced courses are intended for developers that have attended the PIPSI course.

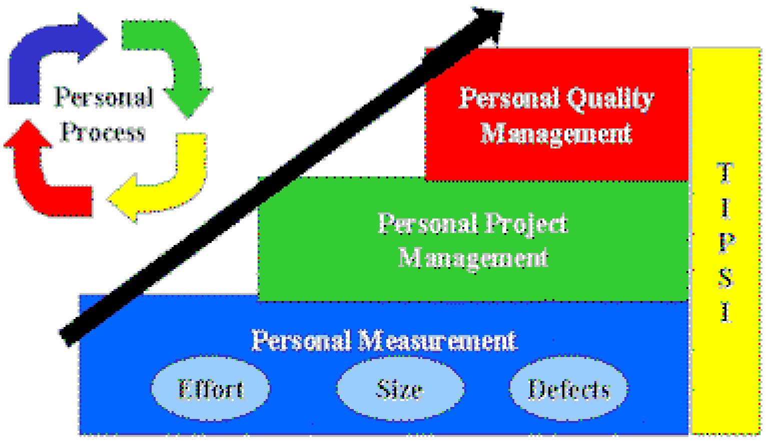
After the training has been attended, PIPSI should be adapted to the context of a company or group of developers to achieve effective day-to-day use. In this phase the company and PIPSI developers collaborate to define the needed adaptation.

PIPSI Components

PIPSI training materials all address, at different levels of detail, a number of concepts that we group under the heading of a number of components.

PIPSI defines a **Personal Process** to be used and adapted by the individual. The three main phases of the process are PreBuild, Build and PostBuild. The Personal Process is the foundation to define Personal measurement and to use specific techniques and methods, Personal Project Management and Personal Quality Management.

Personal Measurement introduces personal measures (effort, size and defects) to monitor and improve the personal process. In more detail, the measures are. Effort (per project, and per phase). Size of the product delivered (Lines of code, or other can be used if more suitable). Defects, total number of defects, defects injected and removed per phase, defect density, defect removed via preventive quality activities. All measures are considered in two variants, as estimated at the beginning of the project, as actual at the end of the project.



Personal Project Management introduces activities and techniques to estimate the size and effort of a project, plan a project, schedule it, track its advancement. First the size of a project is estimated, using analogy or decomposition. Then effort is estimated, using analogy or past productivity figures. Finally effort is allocated to phases of the process. As the project proceeds, actual effort is tracked, and compared with the estimates, with replanning taking place as required.
Personal Quality Management deals with defect collection and analysis to track the quality of a project. Further, it introduces design and code reviews to improve the quality of a project.

Conclusion

PIPSI is an evolution of the PSP, aimed at solving of the problems raised by using the PSP in practice, in industry. PIPSI simplifies the concepts from the PSP and packages them in a shorter, more modular training format that better satisfies the needs of busy SMEs. PIPSI defines a more simple and adaptable process, that addresses quality and project management using metrics to monitor their effectiveness. A tool supports all activities. Transition to day to day use in industry is envisaged and supported.

References

[1] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison Wesley, 1999.

[2] El Emam, K., B. Shostak and N.H. Madhavji. Implementing Concepts from the Personal Software Process in an Industrial Setting, *Proceedings of Software Process 1996*, IEEE CS Press, Brighton, UK.

[3] Ferguson, P., W.S. Humphrey, S. Khajenoori, S. Macke, and A. Matvya. Results of Applying the Personal Software Process, *IEEE Computer*, May 1997.

[4] Hayes W., and J. Over, The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers, Technical report SEI-97-TR-001, December 1997.

[5] Humphrey, W.S., *A discipline for Software Engineering*, Addison-Wesley, 1995.

[6] Humphrey, W.S., Lovelace M., Hoppes R. *Introduction to the Team Software Process*, Addison-Wesley, 1999.

[7] Humphrey, W.S., Using a Defined and Measured Personal Software Process. *IEEE Software*, pages 77-88, May 1996.

[8] Morisio M., Applying the PSP in industry, *IEEE Software*, accepted for publication, 2000.

[9] Taylor A., IT Projects: Sink or Swim, *The Computer Bulletin*, January 2000.

A Tool to Support the Capture of Individual Process Data

Rory O'Connor, Howard Duncan
Dublin City University, Ireland

Gerry Coleman
Dundalk Institute of Technology, Ireland

Marisa Escalante, Cliona McGowan
European Software Institute, Spain

David Escala, Maurizio Morisio
Politecnico di Torino, Italy

Christophe Mercier
AFTI, France

Yingxu Wang
IVF, Sweden

Introduction

Software developers and managers have faced the problem of producing quality software since the beginning of the software age. Many people have studied the software quality problem and have proposed solutions, including better testing, better project planning, better practices, better programming environments and many other factors that potentially affect the development of software. We can categorize these different solutions into two groups: Firstly, solutions that focus on software development as a group effort and secondly, solutions that focus on the individual software developer. Some of the many suggestions that involve groups of software developers include: the Capability Maturity Model, clean room development, software quality assurance groups and formal technical review groups. These organisational level methods help improve the quality of the software, however they may not be enough.

Many user needs in SPI are not fully catered for through existing software process models. In particular, there is an absence of support for process improvement at the individual level, although according to a recent survey of European software organisations 83% of companies believe that SPI is essential for future success and 87% of companies believe that SPI can significantly improve software quality [1]. However time, cost and lack of knowledge are seen as the main barriers to SPI usage. In Europe to date, SPI has focused at the organisational level. As a result a number of issues have been identified in the European software industry at this level [2]:

- There is no clearly defined framework at the individual engineer level to enable process improvement
- Software engineers work with an increasing array of tools in a range of development environments.
- The SME needs a suitable method to improve their software engineers' capability for developing software of higher quality on schedule and to budget.
- The SMEs need a set of personal process training material that is suitable to the European software industry.

At the personal level, the support available to the individual software engineers in performing their software engineering activities a matter of concern. This is often referred to as the Personal Software Process (PSP), a term coined by Watts Humphrey [3]. The main purpose of providing process resources at this level is [4]:

- To support the automation of mundane tasks and activities.
- To monitor the personal performance of individual software engineers by reporting on the status of their personal software engineering activities.
- To provide guidance and help to software engineers in improving their individual software processes.

The PSP has been successfully used within organisations already using SPI methods, where the culture of quality and process discipline is strong. However, it has not been so successful in smaller or less disciplined organisations. Some of the issues associated with the PSP are:

- There is a significant training period (excess of two weeks).
- It was developed for an academic environment, thus it is difficult to customise for industrial use.
- Many companies involved in PSP training have failed to implement it in a industrial context.
- Bureaucratic overhead makes it more difficult to sustain in an industrial setting
- Lack of an adequate support tool for data gathering.

PIPSI Approach

The IPSSI (Improving Professional Software Skills in Europe) project [5] is an ESSI funded project which aims to provide a process improvement framework for use by individual software engineers working in European SMEs. The focus of the project is on improving individual software engineering skills thus generating bottom-up improvement. Companies can experiment with SPI by sending individuals on IPSSI training courses and then monitoring its implementation. By training individuals in this way, costs are reduced. At the heart of the IPSSI initiative is the PIPSI (Process for Improving Programming Skills in Industry) approach, whose aim is to present the techniques in a way that makes them more attractive and more easily used in small and medium-sized organisations and development teams.

PIPSI focuses on two areas of concern to the software engineer – Personal Project Management and Personal Quality Management. Both of these are supported by an estimating process and the use of appropriate metrics for measuring past success in estimating and providing a basis for future planning. Each program worked on is treated as a 'project' by the software engineer, who starts with a planning and estimating phase before starting work on the program. As the work progresses detailed metrics are kept, which are used both to monitor progress and to build up a history. These metrics are personal to the programmer at all times, and are not available to management for productivity assessment or similar purposes. The entire personal process is cyclical, success in one personal project building on success in previous projects. As software engineers become comfortable with the concepts, new and more sophisticated techniques can be introduced to improve both Project Management and Quality Management.

PIPSI is built around the following key concepts:

- A Personal Process to be used and adapted by the individual. The three main phases of the process are PreBuild, Build and PostBuild. The Personal Process is the foundation needed to define Personal measurement and to use specific techniques and methods, Personal Project Management and Personal Quality Management.
- Personal Measurement which introduces personal measures (effort, size and defects) to monitor and improve the personal process.
- Personal Project Management which introduces activities and techniques to estimate the size and effort of a project, plan a project, schedule it and track its advancement.
- Personal Quality Management which deals with defect collection and analysis to track the quality of a project. Further, it introduces design and code reviews to improve the quality of a project.

The focus of PIPSI is on bottom-up process improvement. Figure 1 illustrates the three elements of personal software engineering - defining a personal process, personal project management and personal quality management.

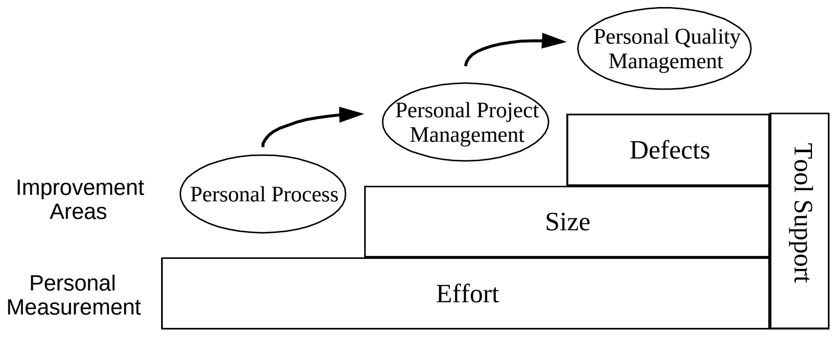


Fig.1 : PIPSI Structure

The entire model is buttressed and controlled through the use of measurement. By collecting data on their own performance, software engineers learn about how they develop software. The measures help them understand the fundamental relationship between size and effort and, through this understanding, enable them to improve their estimating abilities. Furthermore, by gathering data on their defect rates they witness how using practices such as personal code reviews and the use of checklists will allow them to produce higher-quality software products. The measures provide information on performance, information can then lead to process improvement and process improvement can lead to the production of better quality software on time. Finally collecting performance data continuously moves developers from defining their own development process, through managing it to optimising it.

Through PIPSI training, developers complete programming tasks on which they collect increasing quantities of data. Early exercises capture effort measures. Subsequent exercises gather size data whilst the concluding exercises capture defect and quality measures. This is hugely empowering for both the programmer and the organisation as a whole. Programmers are now in a position where they can provide the project manager with achievable deadlines and the project manager can develop more accurate and predictable delivery schedules. The final element of PIPSI is that of personal quality management. As developers complete PIPSI program using exercises, they collect data on the defects injected into those programs. This process illustrates in which development phases they inject and remove defects. Furthermore, the defects are categorised by type thus allowing a causal analysis to be performed which can then lead to defect prevention. PIPSI focuses on proven quality control mechanisms such as design and code reviews which enable developers to remove defects earlier in the development process. This achieves the twin objectives of removing defects at the front end of the development cycle where they are cheaper and easier to fix and, as a corollary, means testing time is more focused as fewer defects are escaping into test.

This process information highlights the developers strengths and weaknesses and empowers them to make the necessary process improvement adjustments. We believe the provision of such a tool will ensure the 'buy-in' of training participants and subsequent continued usage of the PIPSI disciplines.

PIPSI Support Tool

Empirically based process improvement frameworks such as PSP and PIPSI focus on the individual software engineer and require them to record data about time spent programming, the defects they find in their software and size of the software, etc. The PSP as described by Humphrey is a manual process. The engineer records, transfers and analyses he data all on paper forms. After many projects, the engineer accumulates a large paper database of their historical data.

Feedback from PSP training programmes in Ireland [6] has suggested that the absence of a support tool, to simplify the recording and analysis of the data produced is one of the major barriers to continued usage of the methods. Developers tire of recording data on paper forms and eventually usage of the disciplines peters out. There is also corroborating evidence from the USA. For example, [7] and [8] report on experiences of a two year PSP study which questioned the quality of the data recorded. They found that there was significant data quality issues with manual PSP, for example, not all defects were recorded because the overhead in recording was too expensive.

These experiences and observations from the PSP, lead the IPSSI project consortium to consider designing an automated, empirically based personal process improvement tool to support the data collection and analysis requirements of the PIPSI approach to process improvement. Therefore as part of the IPSSI project, a tool set, which consists of data gathering and data analysis tools for use in a web-based environment, has been developed. The main requirements of this tool were that it enables measures to be collected as a simple complement to the development process and provides for analyses of the data collected to provide the developer with important process feedback.

The PIPSI tool was developed in order to support the individual developer using the PIPSI approach and is designed to support two main functions (as illustrated in figure 2): Data Capture - simple and easy recording of data such as time, size and defects, and Data Analysis - automatic analysis of collected data to generate aggregate project data in textual and graphical form. In addition, there were also a number of constraints placed on the development of the tool:

- All data gathered and analysed during a process should be stored in a private database.
- The developer can optionally elect to anonymously submit gathered data to a central database.
- The tool should be platform independent.
- The developer (user) should be able to work in a stand-alone manner, i.e. no network connection required.

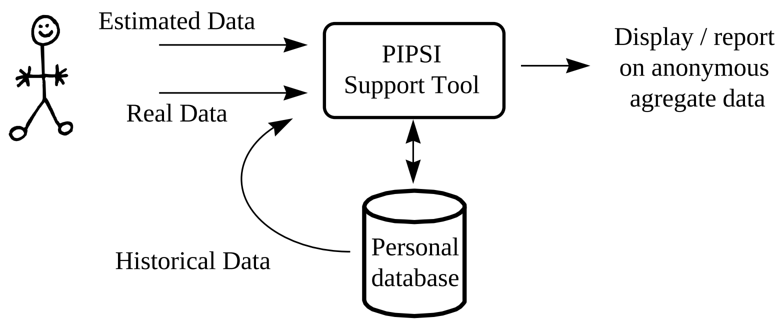


Fig.2 : PIPSI Tool Usage

The PIPSI model follows the three elements of personal software engineering; defining a personal process, personal project management and personal quality management. These are represented in the tool by three main levels:

- Recording basic data such as: size, effort and defects.
- Advanced review techniques and statistical analysis of derived measures including estimated size, real size and defects found in review.
- Expert topics such as design methods, earned value tracking, design errors found, schedule and task planning, productivity and design quality.

Tool Design and Implementation

The tool was designed with a view to having three possible configurations in terms of which component(s) may be installed on which host machine. The three typical configurations which are envisaged are:

- **Standalone desktop configuration** - Where the developer works on a standalone machine which has all components installed on a single machine, say a typical end user desktop PC. The developer has no requirement to access or share their data.
- **Network (Intranet) configuration** - Where the developer is working in an environment which is connected to a company network (Intranet). Here the tool may be accessible locally or over the network via the web browser, with the database either installed on the developers local machine or anonymously submitted to a central database.
- **Network (Internet) configuration** - This is a variation of a standard network configuration above. Here it is possible to access the tool (via a web browser) which is located on a central machine controlled by the IPSSI project consortium. Once again, all data is submitted anonymously.

The architecture of the tool is shown in figure 3. The user interface, which operates within a standard web browser and provides facilities for data gathering, analysis and submission to a local database. Upon completion of a project, the user may choose to anonymously submit their data to a central database where aggregate data can be further collected and analysed.

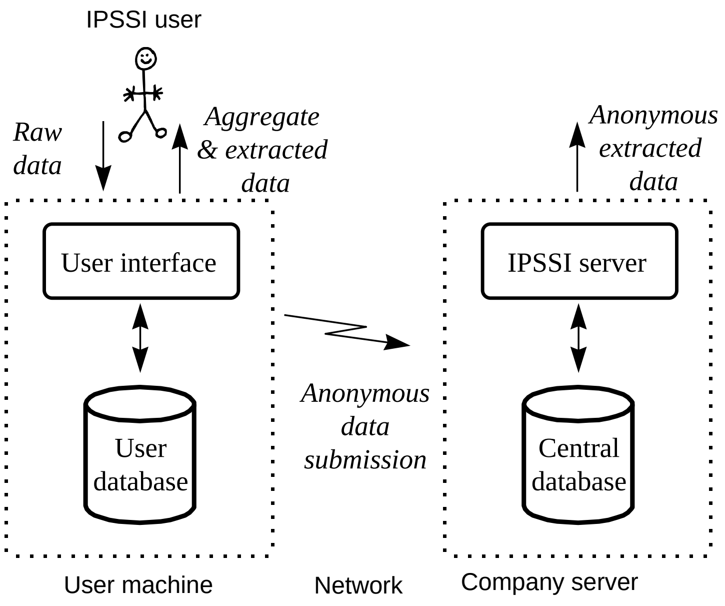


Fig.3 : PIPSI Tool Architecture

The PIPSI support tool was implemented using Active Server Pages which interact with a database system. This provides for a light weight implementation which facilitated the three configurations described above and also provided a flexible approach to system implementation.

The web pages of the system are a code mix of VBScript and standard HTML. The VBScript of the requested web active server page is interpreted by the web server, which generates HTML code which in turn is sent to the clients browser and is displayed like a pure HTML page. The underlying PIPSI data is held in a standard Microsoft Access 2000 database , which is accessed by the Active Server Pages via ODBC connection to the database.

In addition to the design considerations previously described, was the issue of localization of the user interface in terms of the language which is presented on the web browser (user interface). This issues was approached by allowing the user (software developer) to choose at tool startup which language they wished to work in. The text of the user interface is held in a series of files from which the Active Server Pages extract the text to be displayed. Currently the system supports the following languages: English, French, Spanish, Dutch, German, Italian, Swedish, Danish, Finnish, Greek and Portuguese.

Current Status

The PIPSI training programme has been developed in a highly iterative manner by a consortium of both academic and commercial organisations. Initially a set of PIPSI programme trials have been conducted using both academic and industrial participants. The objectives of the initial PIPSI trials [9] using undergraduate students were twofold. Firstly, to teach them the benefits of having a defined and measurable software process and secondly to get some preliminary feedback on the PIPSI training programme. All of the graduates commented favorably on the approaches taken. Comments ranged from "It's something I'd never thought of before", to "Prior to this I had no knowledge of where I was spending my time" to "Up to now I have been a 'hacker' who started coding as early as possible. I have now started to concentrate more of my effort in design and I can already see the benefits". Whilst the results from the study may not readily show this, the study group are now adopting a much more professional approach to developing software and have been convinced of the benefits of following a defined process. These trials also provided the necessary feedback on the PIPSI training material which was used in the evolution of the training programme. A second set of trials were conducted using a small group of industrial participants. The primary objective of these trials was to validate both the structure and content of the PIPSI training programme prior to a commercial launch of PIPSI.

Currently the IPSSI consortium is in the process of running a series of PIPSI training courses in several European locations. These courses are a full commercial version of the PIPISI course and are being attended by industrial participants from an number of European SMEs. To date these courses have taken place in Dublin (Ireland), Bilbao (Spain) and Turin and Milan (Italy), with a number of other PIPSI courses currently planned. The PIPSI support tool has been used during these courses and has gathered a set of PIPSI data which is currently being analysed by the IPSSI project consortium.

However, there are two factors which will permit a more complete and effective evaluation of PIPSI. Firstly, if PIPSI is to succeed, it must be applied in an industrial context. The training and the disciplines are designed to allow individuals and companies achieve meaningful software process improvement. The ultimate benefits should accrue when the skills acquired in training are applied by the practitioners in their workplace and it is this which will truly test the value of PIPSI. Secondly, when the initial set of commercial courses are completed, a larger amount of data will be available for study. As these courses are aimed at industrial practitioners, it is expected that motivation and completion rates will be significantly higher than the voluntary trials which have taken place to date.

Improving the quality of software products is a very important goal for both companies and developers. Ultimately, it is through applying PIPSI disciplines in their own daily work that developers can become convinced of its benefits.

Acknowledgments

This research was supported by the European Commission as ESSI project 27453.

The authors gratefully acknowledge the value gained from discussions regarding this work with their colleagues and research partners: AFTI (France), Dublin City University (Ireland), European Software Institute (Spain), Politecnico di Torino (Italy) and IVF (Sweden). More information about the IPSSI project is available from: http://www.compapp.dcu.ie/ipssi/ipssi.html

References

[1] "The SPIRE Handbook", Centre for Software Engineering, Ireland, 1998.

[2] Y.Wang, H.Duncan, M.Kartinnen, H.Sjostrom and P.Kokeritz, „IPSSI - A European Methodology on PSP“, Proceedings EuroSPI-99, Finland, 1999.

[3] W.Humphrey, „Introduction to the Personal Software Process“. Addison Wesley, 1997.

[4] S.Zahran, „Software Process Improvement - Practical Guidelines for Business Success“, Addison Wesley, 1998.

[5] http://www.compapp.dcu.ie/ipssi/ipssi.html

[6] P.O'Beirne and J.Sanders, „Personal Software Process: Does the PSP Deliver its promise?“, Proceedings of Inspire ‘97, Gothenburg, Sweden, 1997.

[7] A.Disney and P.Johnson, „Investigating data quality problems in the PSP“, Proceedings of the ACM SIGSOFT Sixth International Symposium on the Foundations of Software Engineering, Florida, USA, 1998.

[8] C.Moore, „Personal Process Improvement for the Differently Disciplined“, Proceedings of the International Conference on Software Engineering, Los Angeles, USA, 1999.

[9] G.Coleman and R.O'Connor, „Power to the Programmer“, In Proceedings of 11th European Software Control and Metrics conference, Germany, 2000.

Deriving Personal Software Processes from Current Software Engineering Process Models

Yingxu Wang
Dept. of Electrical and Computer Engineering
University of Calgary
2500 University Drive, Calgary, Alberta, Canada T2N 1N4

IVF Centre for Software Engineering
Argongatan 30, S-431 53, Molndal, Gothenburg, Sweden
Yingxu.Wang@acm.org
?

Graham King
Design and Advanced Technology Research Centre
Southampton Institute
East Park Terrace, Southampton, SO14 0YN, UK
Graham.King@solent.ac.uk

Howard Duncan
Dept. of Computer Science
Dublin City University
Dublin 9, Ireland
Howard@compapp.dcu.ie

ABSTRACT: Hierarchically, software engineering processes may be classified at organizational, project and personal levels according to users of the processes. There are independent personal process models such as PSP and IPSSI. A new approach to establish a personal process model is to derive it from a software engineering process reference model. A generic Software Engineering Process Reference Model (SEPRM) has been developed for enabling flexible process models to be derived at personal, project, and organizational levels. This paper describes how a personal software process model, the PSPM, is derived from SEPRM by incorporating the advantages of existing process models and international standards. Practical guidelines for tailoring SEPRM in the process dimension and for determining target capability levels in the capability dimension are provided for personal process users.

Key words: Software engineering, software process, personal process, model, development, plan, quality assurance

1. Introduction

A number of software engineering standards and models have been developed in recent decades, such as CMM [1 - 4], ISO 9001 [5], Trillium [6], BOOTSTRAP [7], ISO/IEC 12207 [8], ISO/IEC TR 15504 (SPICE) [9]. Software engineering processes can be classified at organizational, project, and personal levels hierarchically according to users of the processes. A personal software process model was first developed by Watts Humphrey known as PSP^{SM} [10], which is designed for individual programmers and software engineers in the environment of small organizations or small projects. A counterpart of PSP developed in Europe is the IPSSI personal process model [11].

Logically, a personal process model may be derived, in context, from an organization's software engineering process reference model. A generic Software Engineering Process Reference Model (SEPRM) has been developed [12] that provides a superset of software engineering processes by integrating major current process models. This paper describes how a personal software process model, the PSPM, is derived from SEPRM by considering the features of a software development project. Practical guidelines for tailoring SEPRM in the process dimension and for determining target capability levels in the capability dimension are provided for personal process users.

2. An Integrated Software Process Model - SEPRM

A software engineering process reference model (SEPRM) [12 - 16] is developed that integrates major current process models and standards such as CMM, ISO 9001, BOOTSTRAP, ISO 15504 (SPICE), and ISO 12207. SEPRM identifies a superset of software engineering processes, and models them in 3 process subsystems, 12 process categories, 51 processes, and 444 base process activities (BPAs). A high-level hierarchical structure of the SEPRM framework is shown in Figure 1.

2.1 The SEPRM Process Dimension

This subsection describes SEPRM processes for software development, management and organization. A hierarchical structure of SEPRM processes is provided and tailorability of the SEPRM process dimension is discussed.

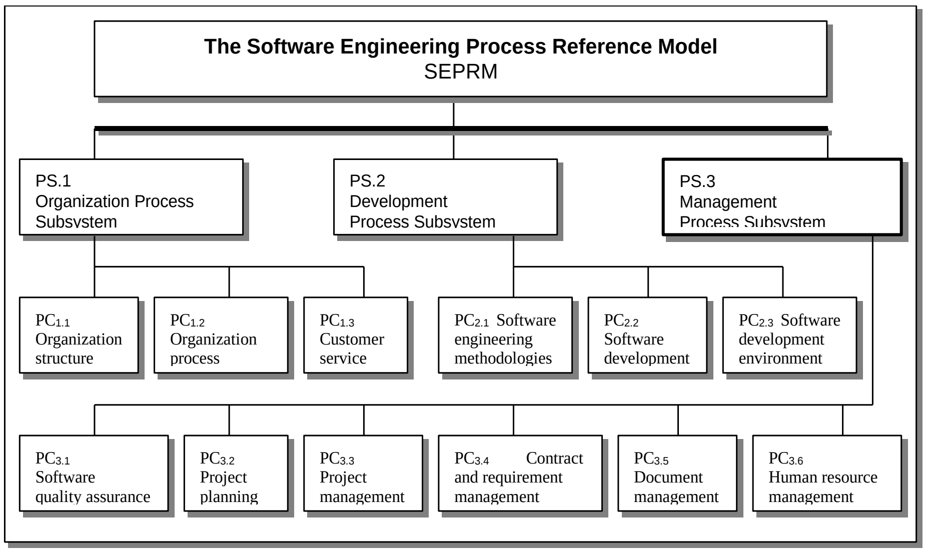

The Software Engineering Process Reference Model
SEPRM
PS.1 Organization Process Subsystem
PS.2 Development Process Subsystem
PS.3 Management Process Subsystem
PC1.1 Organization structure
PC1.2 Organization process
PC1.3 Customer service
PC2.1 Software engineering methodologies
PC2.2 Software development
PC2.3 Software development environment
PC3.1 Software quality assurance
PC3.2 Project planning
PC3.3 Project management
PC3.4 Contract and requirement management
PC3.5 Document management
PC3.6 Human resource management

Figure 1. Hierarchical structure of SEPRM

2.1.1 Structure of the SEPRM process model

The SEPRM software engineering model, as shown in Table 1, consists of 51 processes and 444 BPAs. All SEPRM processes are classified into three process subsystems known as organization, development, and management.

Table 1. The SEPRM Process Model

ID.	Subsystem	Category / Process	Identified BPAs	Adaptable for personal process
1	Organization		81	
1.1		Organization structure processes	13	
1.1.1		Organization definition	7	
1.1.2		Project organization	6	
1.2		Organization processes	26	
1.2.1		Organization process definition	15	
1.2.2		Organization process improvement	11	
1.3		Customer service processes	39	
1.3.1		Customer relations	13	
1.3.2		Customer support	12	
1.3.3		Software/system delivery	11	√
1.3.4		Service evaluation	6	
2	Development		115	
2.1		Software engineering methodology processes	23	
2.1.1		Software engineering modeling	9	
2.1.2		Reuse methodologies	7	√
2.1.3		Technology innovation	7	
2.2		Software development processes	60	
2.2.1		Development process definition	12	
2.2.2		Requirement analysis	8	√
2.2.3		Design	9	√
2.2.4		Coding	8	√
2.2.5		Module testing	6	√
2.2.6		Integration and system testing	7	√
2.2.7		Maintenance	10	√
2.3		Software engineering infrastructure processes	32	
2.3.1		Environment	7	
2.3.2		Facilities	15	
2.3.3		Development support tools	4	
2.3.4		Management support tools	6	
3	Management		248	

3.1		Software quality assurance (SQA) processes	78	
3.1.1		SQA process definition	17	
3.1.2		Requirement review	5	√
3.1.3		Design review	4	√
3.1.4		Code review	3	√
3.1.5		Module testing audit	4	√
3.1.6		Integration and system testing audit	6	√
3.1.7		Maintenance audit	8	√
3.1.8		Audit and inspection	6	
3.1.9		Peer review	10	
3.1.10		Defect control	10	√
3.1.11		Subcontractor's quality control	5	
3.2		Project planning processes	45	
3.2.1		Project plan	20	√
3.2.2		Project estimation	7	√
3.2.3		Project risk avoidance	11	
3.2.4		Project quality plan	7	
3.3		Project management processes	55	
3.3.1		Process management	8	
3.3.2		Process tracking	15	
3.3.3		Configuration management	8	√
3.3.4		Change control	9	
3.3.5		Process review	8	
3.3.6		Intergroup coordination	7	
3.4		Contract and requirement management processes	42	
3.4.1		Requirement management	12	
3.4.2		Contract management	7	
3.4.3		Subcontractor management	14	
3.4.4		Purchasing management	9	
3.5		Document management processes	17	
3.5.1		Documentation	11	√
3.5.2		Process database/library	6	
3.6		Human resource management processes	11	
3.6.1		Staff selection and allocation	4	
3.6.2		Training	7	
Total	3	51	444	19

2.1.2 Tailorability of the SEPRM process model

As introduced in Section 1 software engineering processes may be classified at organizational, project, and personal levels according to users and purposes of the process model. The tailorability provided in SEPRM enables flexible personal process models to be derived. Referring to Table 1, the adaptability of each process in SEPRM has been shown by '√ ,' indicating that a specific process is suitable for using at personal process level.

There are 19 processes for constructing a personal software process model with a configuration of development, software quality assurance, and project management processes. Detailed discussion of the derived personal process model will be provided in Section 3.

2.2 The SEPRM Process Capability Dimension

In seeking technologies for software process capability measurement, three types of process capability scales have been classified [12], such as:

- Pass-threshold based (as that of ISO 9001)
- Management-attributes oriented (as those of CMM and ISO 15504)
- Process oriented (as that of SEPRM)

The SEPRM process capability model adopts the third type of process capability scale that directly rates the performance of a process within three contexts: organization, project, and individual.

This subsection describes the SEPRM process capability dimension that will be used as a measurement scale for process assessment and improvement. SEPRM adopts a process capability model [12] as shown in Table 3 with six defined process capability levels, raging from Level 0 to Level 5, known as Incomplete, Loose, Integrated, Stable, Effective, and Refining.

Table 3. The SEPRM Process Capability Model

Capability Level (CL[i])	Description	Process Capability Criteria			BPA Average Performance Threshold
		Organization Context	Project Context	Individual Context	
CL[0]	Incomplete	C[0,1] No process reference model	C[0,2] No defined and repeatable process activities	C[0,3] Ad hoc	0-0.9
CL[1]	Loose	C[1,1] There are defined processes to some extent	C[1,2] There are limited process activities defined and conducted	C[1,3] Varying	1.0-1.9
CL[2]	Integrated	C[2,1] There is an established process reference model	C[2,2] There are relatively complete process activities defined and aligned to organization's process reference model	C[2,3] Generally process-based	2.0-2.9
CL[3]	Stable	C[3,1] There is a repeatable process reference model	C[3,2] There are complete process activities derived from organization's process reference model	C[3,3] Repeatedly process-based	3.0-3.9
CL[4]	Effective	C[4,1] There is a proven process reference model	C[4,2] - There are completed process activities derived from organization's process reference model - Performances of processes are monitored	C[4,3] Rigorously process-based	4.0-4.5
CL[5]	Refining	C[5,1] There is a proven and refined process reference model	C[5,2] - There is a completed derived process model - Performances of processes are quantitatively monitored and fine-tuned	C[5,3] Rigorous and optimistic process-based	4.6-5.0

In Table 3, a set of defined criteria for rating the capability of a process is provided, where an index *C[level,j]* with *level = 0,...,5* and *j = 1,2,3,* indicates a process capability criterion in the organization, project, or individual context. Table 3 shows that a software engineering process as a basic unit can be assessed in the organization, project, and individual contexts against the six-level process capability criteria. In order to relate the process capability criteria to the performance of BPAs in a process, there is another criterion for assessing a process: the average performance of the BPAs.

Table 3 shows that there are four criteria that a process has to fulfill in order to reach a specific capability level. The first three are oriented to a process as a whole; the last one is oriented to the BPAs contained in a process. Therefore, the capability of a given process is determined by the maximum level *i* that a process achieved for fulfilling all four criteria for that level.

The SEPRM process assessment results are reported at the six-level scale plus a decimal value. This means it has the potential to distinguish the process capability at tenth-sublevels. This feature enables a software development organization or an individual programmer to continuously monitor and fine-tune its process capability by a series of assessments.

3. Deriving a Personal Software Process Model from SEPRM

As described in Section 2, SEPRM is developed as an overarching software engineering model by integrating the advantages of major current process models. Conventional personal process models were developed specifically and independently. A new approach to establish personal software process is deriving it from an existing software engineering reference model. This section describes how a personal software process model, the PSPM, is derived from SEPRM, and how target capability levels of PSPM processes may be determined for individual software engineers.

3.1 Guideline for Deriving a Personal Process Model

A personal software process model may be derived on the basis of features of a software development project. SEPRM is developed as a generic process model for providing a comprehensive process platform for a software organization or for individuals. Thus, various process models may be derived from SEPRM for different projects either for an individual or a team.

In small software project planning, the first thing that individuals need to do is to derive the project's personal process model. The personal project process model will serve as a blueprint for organizing activities that are going to be enacted within the scope and life-cycle of the project, including technical, managerial, and supporting activities.

A checklist of factors for consideration in deriving a personal process model from SEPRM is shown in Table 4. Given all factors weighted by high (H), medium (M), or low (L), the type of personal process model can be determined according to Expressions 1 and 2.

Table 4. Determining type of personal process model for a project

No.	1.1.1.1.1.1 Project Factor	Weight			Score
		High (5)	Medium (3)	Low (1)	
1	Size		✓		S1 = 3
2	Importance	✓			S2 = 5
3	Difficulty	✓			S3 = 5
4	Complexity	✓			S4 = 5
5	Domain knowledge requirement	✓			S5 = 5
6	Experience requirement	✓			S6 = 5
7	Special process needed		✓		S7 = 3
8	Schedule constraints	✓			S8 = 5
9	Budget constraints			✓	S9 = 1
10	Other process constraints	✓			S10 = 5
Total		35	6	1	**1.1.1.1.1.2**

Assuming that S_i is the ith weight for factor i and n is the total number of factors, the average score, S, or the level of requirement for a derived model is defined as:

$$S = 1/n \sum_{i=1}^{n} S_i \qquad (1)$$

According to the average score S, the type of derived model can be estimated as follows:

$$\begin{array}{ll} S > 4.0, & \text{a \textbf{complete} (C) personal process model is needed} \\ S < 2.5, & \text{a \textbf{light} (L) personal process model is needed} \\ \text{otherwise,} & \text{a \textbf{medium} (M) personal process model is needed} \end{array} \qquad (2)$$

For instance, applying Expression 1 to the sample weights of the ten factors as shown in Table 4 results in an estimated average score $S = 4.2$. According to Expression 2, the personal process model needed for the given project will be a complete process model. This requirement provides a good representation of the nature of the given project with weighted characteristics as shown in Table 4.

3.2 The Derived Personal Software Process Model (PSPM)

When the average characteristic score of a project, S, is obtained according to the guideline provided in Sections 3.1, a specific personal process model can be derived by selecting the processes marked suitable for personal process use as provided in Table 1. For example, according to the project's average characteristic score derived above, a complete personal process model, PSPM, may be derived from SEPRM for individuals working in small organizations and on small projects as shown in Table 5.

It is noteworthy that in PSPM, as shown in Table 5, there is a parallelism between the development processes (No. 3 - 8) and the management processes (No. 9 - 14). This indicates that for each of the development process there is a counterpart for management and quality assurance. These parallel personal processes are the core of the PSPM for ensuring high quality software.

Table 5. A personal software process model derived from SEPRM

No.	Subsystem	Category	1.1.1.1.1.3 Process	Identified BPAs
	Organization		1	
		Customer service		
1			Software/system delivery	11
	Development		7	
		Software engineering methodology		
2			Reuse methodologies	7
		Software development		
3			Requirement analysis	8
4			Design	9
5			Coding	8
6			Module testing	6
7			Integration and system testing	7
8			Maintenance	10
	Management		11	
		Software quality assurance (SQA)		
9			Requirement review	5
10			Design review	4
11			Code review	3
12			Module testing audit	4
13			Integration and system testing audit	6
14			Maintenance audit	8
15			Defect control	10

		Project planning		
16			Project plan	20
17			Project estimation	7
		Project management		
18			Configuration management	8
		Document management		
19			Documentation	11
	Total		19	152

Users may further tailor the PSPM for a Light (L) or Medium (M) personal process model based on specific project needs. The implementation of the selected processes will be on the basis of a set of base process activities, the BPAs, which is provided in [12, 14]. An organization may tailor the BPAs and provide additional activities for a process on the basis of any special requirements and the features of the software project.

3.3 Determination of Target Capability Levels According to SEPRM Benchmarks

Although the conventional goal-based process assessment and improvement technologies have been widely accepted, their philosophy of "the higher the better" has been questioned in practice. It is found that the determination of target capability levels for a personal process tends to be virtual, infeasible, and sometimes overshoot the target [12]. Benchmark-based process assessment and improvement provides a new approach to adaptive and relative process improvement based on a philosophy of "the smaller the advantage, the better [12]." According to the benchmark-based process improvement method, the target capability levels of personal software processes may be set relative to the benchmarks of the software industry or a specific sector, rather than to the virtually highest capability level as in a goal-based process model.

Based on the benchmarks of the SEPRM process reference model as shown in Figure 2, the target capability levels of the SPSM personal processes can be divided into three categories such as:

- Basic level
- Competitive level
- Advanced level

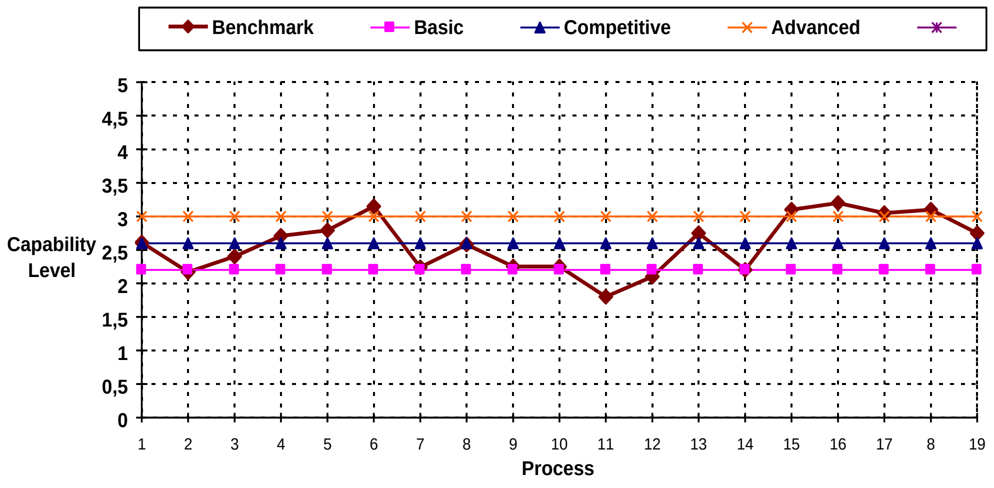


Figure 2.
Process capability benchmarks and target capability levels

The basic level is the minimum level of process capability that a software engineer should achieve in order to develop quality software according to the SEPRM benchmarks. It is suitable as a target for initial programmers who are in the early stages of personal process establishment and improvement.

The competitive level is an average level of process capability that ordinary software engineers have reached in software development according to the SEPRM benchmarks. It is suitable as a target for the established programmers who pursue a stable personal software process and systematic process improvement.

The advanced level is the highest level of process capability that achieved by the top software engineers according to the SEPRM benchmarks. It is suitable as a target for the experienced programmers who aim at optimizing existing personal processes and producing high quality software for complicated and/or mission-critical systems.

By following the steps provided above, a PSPM personal process model can be derived systematically on the basis of SEPRM, and the target process capability level for each personal process is determined according to SEPRM benchmarks. The benchmark-based process improvement concept is found useful for both personal and organizational process improvement, and has shown impact on empirical software engineering practices.

4. Conclusions

Personal software processes are designed for individual programmers and software engineers in the small organization or small project environment. Conventional personal software process models are developed independently from organizational process models. This paper has provided a new approach to personal process model development by deriving it in context and from a well founded software engineering process reference model.

A generic Software Engineering Process Reference Model (SEPRM) has been introduced that provides a superset of software engineering processes by integrating major current process models. This paper has demonstrated how a personal software process model, the PSPM, is derived from SEPRM, and how the target capability levels of personal processes can be determined for personal process improvement.

References

[1] Paulk, M.C., Curtis, B., Chrissis, M.B. and Weber, C.V. (1993), Capability Maturity Model for Software., Version 1.1, *Software Engineering Institute, CMU/SEI-93-TR-24*, February.

[2] Paulk, M.C., Weber, C.V. and Curtis, B. (1995), *The Capability Maturity Model: Guidelines for Improving the Software Process*, SEI Series in Software Engineering, Addison-Wesley.

[3] Humphrey, W.S. and Sweet, W.L. (1987), A Method for Assessing the Software Engineering Capability of Contractors, *Technical Report CMU/SEI-87-TR-23*, Software Engineering Institute, Pittsburgh, PA.

[4] Humphrey, W.S. (1988), Characterizing the Software Process: A Maturity Framework, *IEEE Software*, March, pp.73-79.

[5] ISO 9001 (1989), *Quality Systems - Model for Quality Assurance in Design, Development, Production, Installation, and Servicing*, International Organization for Standardization, Geneva.

[6] Bell Canada (1994), *TRILLIUM - Model for Telecom Product Development and Support Process Capability (Internet ed.)*, Release 3.0, December, pp. 1-118.

[7] BOOTSTRAP Team (1993), BOOTSTRAP: Europe's Assessment Method, *IEEE Software*, May, pp.93-95.

[8] ISO/IEC 12207 (1995): Information Technology – Software Life Cycle Processes, *International Organization for Standardization, Geneva.*

[9] ISO/IEC TR 15504:1-9 (1999): *Information Technology – Software Process Assessment* - *Part 1-9,* ISO/IEC, Geneva.

[10] Humphrey, W. (1997), *Introduction to the Personal Software Process,* Addison Wesley, Reading, MA.

[11] Wang, Y., H. Duncan, M. Kartinnen, H. Sjostrom and P. Kokeritz (1999), IPSSI - A European Methodology on PSP, *Proceedings EuroSPI-99,* Finland, pp.12.40-12.51.

[12] Wang, Y. and King, G. (2000), *Software Engineering Processes: Principles and Applications,* CRC Press, USA, ISBN: 0-8493-2366-5, pp.1-752.

[13] Wang, Y. and Patel, D. (2000), Comparative Software Engineering: Review and Perspectives, Special Volume on Comparative Software Engineering, *International Journal of Annals of Software Engineering,* Baltzer Science Publishers, Oxford, Vol.10, to appear.

[14] Wang Y., King, G., Dorling, A., Ross, M., Staples, G., and Court, I. (1999), A Worldwide Survey on Best Practices Towards Software Engineering Process Excellence, *ASQ Journal of Software Quality Professional,* Vol.2, No.1, December, pp. 34-43.

[15] Wang, Y., Doling, A., Wickberg, H. and King, G. (1999), Experience in Comparative Process Assessment with Multi-Process Models, *Proceedings of IEEE European Micro Conference (IEEE EuroMicro'99),* Vol. II, IEEE CS Press, Milan, September, pp.268-273.

[16] Wang, Y., I. Court, M. Ross and G. Staples (1996), Towards a Software Process Reference Model (SPRM), *Proceedings of International Conference on Software Process Improvement (SPI'96)*, Brighton, UK, November, pp.145-166.

About the Authors

Yingxu Wang is Professor of Software Engineering in Dept. of Electrical and Computer Engineering at The University of Calgary, and project manager with the Center for Software Engineering at IVF, Gothenburg, Sweden. He received a PhD in software engineering from the Nottingham Trent University / Southampton Institute, UK. He is a member of IEEE, ACM, and ISO/IEC JTC1/SC7, and is Chairman of the Computer Chapter of the IEEE Swedish Section. He was a visiting professor at Oxford University. He is the lead author of a recent book on *Software Engineering Processes: Principles and Applications*, and he has published over 100 papers.

Graham A. King is Professor of Computer Systems Engineering at Southampton Institute, UK. He heads a research center which has a strong interest in all aspects of software engineering. He was awarded a PhD in architectures for signal processing by the Nottingham Trent University. He is a co-author of a recent book on *Software Engineering Processes: Principles and Applications*, and has authored three books and nearly 100 academic papers.

Howard Duncan is a Senior lecturer in the Dept. of Computer Science at Dublin City University. He is an experienced software engineer and project manager. Currently, he is coordinator of a European Commission funded research project IPSSI.

Session 6 - SPI and Measurement

Session Chair:

Tor Stalhane,

SINTEF

Measuring the Success of Software Process Improvement: The Dimensions

Pekka Abrahamsson
University of Oulu, Finland
Pekka.Abrahamsson@oulu.fi

Abstract

Quality managers, change agents and researchers are often troubled in defining and demonstrating the level of success achieved in software process improvement (SPI) initiatives. So far, there exist only few frameworks for identifying the level of success achieved in SPI. Analysis shows that these frameworks do not provide a comprehensive view from all relevant stakeholders involved in SPI. Early results from an ongoing research effort to discover and operationalise success dimensions are reported. Adapted from the project management literature it is suggested that five dimensions characterise the level of success achieved in SPI: (1) project efficiency, (2) impact on the process user, (3) business success, (4) direct operational success and (5) process improvement fit. Results from an empirical analysis are reported where 23 change agents evaluated the relative level of importance of each dimension. Early results indicate that change agents valued the process user satisfaction the most and the process improvement fit the least. This finding confirms the need of having various stakeholders and dimensions acknowledged in a framework that is used to measure the overall success of an SPI initiative.

Keywords: software process improvement, success, measurement, success dimensions, multidimensional constructs

Introduction

In recent years many excellent books on practical software process improvement (SPI) (e.g. [8, 10, 21]) have been written. Still change agents, quality managers, process owners and researchers are often troubled in defining the level of success achieved in SPI.

There does not exist a universal framework with which the success of a project could be measured and assessed [18]. As of today, this argument remains true in engineering like projects as well as process improvement projects. However, some authors (e.g. [15]) call for learning from the previous successes while others (e.g. [1]) call for learning from failures. In order to learn from previous experiences, one needs to be able to separate a success from a failure. SPI research has paid little attention to systematic study about the conditions under which SPI initiatives vary in their results [4].

The issue of success of an SPI initiative is deemed to be a problematic one since we all have experiences on projects that initially appeared to be failures (e.g. projects were not finished in time) but later on turned out to be successes (e.g. fault rate was dramatically dropped).

It has been acknowledged that software measurement is essential in the improvement of software processes and products since if the process (or the result) is not measured the SPI effort could be addressing the wrong issue [21]. However, the issue of process measures is generally considered to be a level four issue in the widely used Capability Maturity Model (CMM[1]). Still, vast majority of the organisations undertaking SPI activities are in level one or two in respective scale. For a such type of an organisation the undertaking of process measures could prove to be a difficult task since in spite of extensive literature on software measurement (see e.g. [16] for overview) companies are facing serious problems initiating even the simplest metrics programs [9].

The issue of direct measurement of a success of any particular SPI initiative is not the main objective of this paper however, since a measure developed without thorough understanding of the concept of interest is not a true measure and may result e.g. in serious ambiguities when interpreting results [20]. Therefore, as a step toward a mature measurement of success in SPI the dimensions that can be used to characterise success are introduced. These dimensions are drawn from project management literature and adapted to SPI environment. It is shown that any direct measure of success remains inadequate if other dimensions are not considered. Also, it is demonstrated that the importance of these dimensions vary depending on the stakeholder (e.g. software developer, change agent or manager) evaluating it. In this paper, early results of an empirical test are reported where 23 change agents evaluated the level of importance of each dimension from their point of view. In short, results indicate that a) all process success dimensions were evaluated to be at least moderately important to change agents, b) the process user (i.e., target of change) satisfaction dimension was rated the highest while c) the ability of an SPI initiative to prepare the organisation for future initiatives was ranked the lowest.

This paper is organised as follows: First, a sample selection of studies that address the issue of success in SPI are introduced. Secondly, the concept of success in SPI is considered by (a) defining the stakeholders involved in SPI initiative, (b) introducing the dimensions that characterise success in SPI, and (c) addressing the issue of Characterising the success using the dimensions identified. Thirdly, results from an empirical study are presented together with conclusions and a future research agenda.

Related Literature

El Emam, Goldenson, McCurley and Herbsleb [4] present a multivariate model of conditions that may be used to explain the successes and failures of SPI efforts. Their model is based on data of CMM based improvement initiatives. Their main finding is that the factors distinguishing between successes and failures are the extent the organisation is focused in its improvement effort, organisational politics and the level of management commitment. El Emam et al. address an important issue but by identifying factors that contribute most to the success of an SPI initiative they do not differentiate the type of successes they deal with. Fitzgerald and O'Kane [6] did an action research study and focused on identifying factors that contribute to success in achieving particular CMM maturity level. Their data is drawn from Motorola's Cellular Infrastructure Group in Cork, Ireland. While their findings are of great interest to organisations pursuing process improvement activities using the CMM approach, the concept of success remains vague to the reader. These two studies are examples of a stream of research aiming at identifying critical success factors of SPI initiatives. Others (e.g. [11, 21]) have also addressed the problem.

Recently Reo, Quintano and Buglione [17] addressed the issue of measuring SPI by extending the infrastructure and innovation perspective of ESI's (European Software Institute) Balanced IT Scorecard (BITS) Generic Model. They propose using the model in identifying goals, drivers and indicators that can be further used in aligning improvement programmes with business objectives.

[1] Capability Maturity Model and CMM are service marks of Carnegie Mellon University

Their extended model addresses e.g. people issues (satisfaction and competence) and organisational climate (continuous improvement, organisational learning and innovation) issues aside the traditional technological issues. They emphasize the need for addressing a wider range of issues than directly process related when measuring software process improvement. While providing a comprehensive list of themes that affect e.g. employee satisfaction and competence, they do not address the issue of success directly other than pointing out that in order to be successful one needs to concentrate on these themes.

Kitchenham [12] has written a comprehensive book of measuring SPI. The book aims at explaining how software measurements can be used to support SPI activities. However, while addressing the measurement issues Kitchenham does not take into account other stakeholder positions. She seems to discuss software metrics as sole indicators of success in SPI. For example, process user satisfaction with the altered processes is not discussed. While hard metrics[2] do provide a great deal of support for process improvement they do not give whole-hearted picture of the impact that a particular SPI programme has on the organisation.

Considering Success in SPI

Collins Electronic English Dictionary defines the term 'success' as "the favourable outcome of something attempted". This literary definition gives a lot of room for interpretations and invites confusion if directly operationalised without considering e.g. all viewpoints present in SPI. Therefore, we shall start considering success by introducing all relevant stakeholders that may be involved in SPI programme.

Stakeholders in SPI

There are multiple types of roles that different interest groups play in an SPI project such as the sponsorship role, management role, coordination role, improvement team role (operational role) [21] and the end-user role (the target of change). Within each of these roles there are people involved with different professional and personal aims as well as different ambitions.

Success means different things to different people [7]. The corporate executive acting in the sponsorship role is concerned with the overall organisational business benefits yielding from the process improvement initiative while on the other end of the spectrum the targets of change (e.g. software developers) might be more concerned with the usefulness of the new or modified process expecting to benefit from the new tool, or procedure in a way that enables them do their jobs better. Targets of change as end-users of SPI activities are in a key position when considering the success of SPI. In general, a systems development project is considered to be a failure if the system developed is not used [14]. Similarly in SPI, if the improved process is not used after initial trials, it should be considered as a failure. Any type of software process improvement success measurement should therefore reflect a more of a multidimensional approach reflecting the expectations of senior management as well as targets of change rather than a single dimensional approach. Table PABRAHAM.1 summarizes the role[3] of various stakeholders and their function in an organisational software process improvement programme.

Role	Person(nel)	Function in SPI
Sponsorship	A corporate executive	Authorization of budgets and resources

[2] Hard metrics (e.g. productivity in terms of lines of code per staff month or design size in terms of number of modules in a product) is used here as opposite to soft metrics (e.g. work morale or level of excitement).

[3] Depending on the size of an organization these roles may overlap and be performed by the same person.

Role	Person(nel)	Function in SPI
Management	A steering committee	Provides management guidance and strategies; monitors progress; resolves organisational issues; promotes SPI goals
Coordination	SEPG (Software Engineering Process Group)	Provides coordination; technical guidance; owner of the SPI plan
Operational	Software process improvement team (change agents)	Manages and implements process improvement activities
Object of change	E.g. software developers, testers	Participates in the change; adopts new behaviours or tools; alters their attitudes

Table PABRAHAM.1: Stakeholders typically involved in software process improvement programmes [21]

Success Dimensions

The following introduction of dimensions that can be used to characterise success in SPI is adapted from project management literature where the issue of success has been under debate for over two decades. Project management literature can be seen as a suitable reference literature since most SPI activities are performed as projects with their own budget, timescale, dedicated resources and goals. The empirical part of the study identifies the importance of these dimensions from the viewpoint of change agents. Change agent is viewed here as a person or a group of persons (operational role in SPI, in Table PABRAHAM.1) facilitating and/or responsible for the attempted change effort.

Shenhar, Dvir and Levy [18] have proposed a multidimensional, multicriteria success assessment framework, which in their view reflects different interests and views set by various parties involved in the project. Shenhar et al. provided a set of dimensions that accumulate the total success level of a project. Each of the following dimensions is important in their own right depending on the type of improvement initiative and the extent of change pursued:

D1: Project Efficiency (Meeting Constraints)

According to Shenhar et al. the first dimension reflects a short-term measure that expresses the efficiency with which the project was managed. This classical dimension found in all of the software project management literature in mainly concerned whether the project was (a) finished on time within (b) specified budget limits. This dimension is possible to be measured during the project (Fig. PABRAHAM.1) and the results are fully applicable for further analysis after the project has been completed. Zahran [21] lists a comprehensive list of project related measures such as the development time, slippage, work completed, effort expended, funds expended, etc. These measures can be used as post mortem (as well as ongoing) project analysis to provide feedback on the level of success in terms of project efficiency success dimension. Even though it is important to coordinate and manage an SPI initiative according to good project management principles, as of its own it does not indicate whether process improvement has been successful or not.

D2: Impact on the Process user (object of change)

Shenhar et al. analyse the success in general level within all types of engineering projects and they refer in this case to the customer. In the case of improving software processes the customer is the user of the process (e.g. project; software developer, tester). The level of success is characterised therefore in terms of level of satisfaction with the new process, fulfilment of the needs of the software developers, solving the problem they experienced and whether the improved process actually is used.

The success in this dimension (Fig. PABRAHAM.1) may be seen relatively early in an SPI initiative as process users become more educated about the process improvement initiative. Improved work morale, excitement and positive attitudes are signs of success in the early phases of the project. Process improvement activities (e.g. coaching, information sharing and training) may reinforce process users' professional competence, which in turn are visible signs of success later on in SPI initiative. Measuring the extent this dimension is a success is more problematic than the first dimension where hard numbers are a relatively straightforward issue.

D3: Business Success and D4: Direct Operational Success

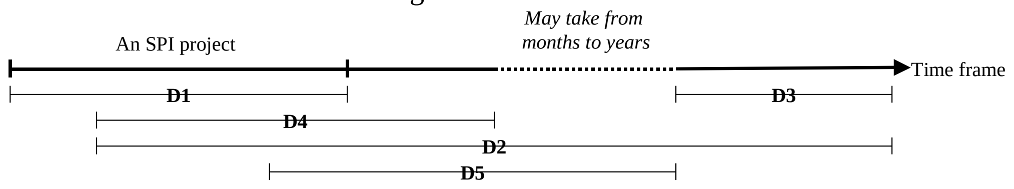
The third dimension in Shenhar's et al. classification addresses the immediate and direct impact the project has on the organisation. In the case of improving software processes this dimension is divided into two sub dimensions concerned with (a) directly process-related aspects and (b) business-related aspects. Zahran [21] includes in his list of process-related measurement many items that directly indicate the level of impact the process improvement project has on the organisation. These measurement examples are e.g. number of change requests, amount of rework, number of process defects, cycle time measures and a maturity level.

Zahran [21] among others argue that one of the major obstacles to the adoption of process improvement is the reluctance of business management to invest in SPI. Zahran points out that this obstacle is raised from the fact that there is a general lack of reliable information on the business benefits of software process improvement. Therefore, by connecting SPI goals to business goals of the organisation the business management is more likely to invest in SPI. Such high-level strategic business goals are e.g. 'gain larger market share', 'reduce time-to-market', 'improve product quality' and 'increase the productivity'. Process improvement programme is an ongoing activity where the immediate results are connected more on the process-related aspects when the business-related results are visible much later on in SPI project's life cycle (Fig. PABRAHAM.1).

D5: Process improvement fit and preparing for the future

The last dimension identified by Shenhar et al. addresses the issue, using their terminology, of preparing the organisational and technological infrastructure for the future. In the case of software process improvement this preparation for the future implies to applicability of an SPI initiative to provide opportunities for future improvement activities. Companies that have had bad experiences in the past with process improvement projects are reluctant to initiate such activities again. The reasons for the failure are manifold and not many times analysed deeper as to what happened. These failing SPI projects did not prepare themselves or the organisation for the future even though they would have had many opportunities to do so. A good example of a post mortem analysis process in software development project that is applicable to SPI initiative as well is introduced in [3].

Indeed, one of the powerful mechanisms for an effective process improvement support infrastructure is its continuously improving characteristic. This requires the establishment of effective feedback mechanisms that ensure direct communication channels between process users, process owners, process groups, project managers and business managers [21]. This in turn enables the organisation to redefine its goals (in respect to SPI programme) and tune the activities depending on the feedback received from various stakeholders. Failure in one specific dimension may prepare the organisation for future challenges but only if it is taken carefully into consideration e.g. in the form of knowledge transfer. On the other hand, a success in one specific SPI action may not necessarily be as relevant as how well the action fits in the larger view of the whole set of activities.



D1: Project efficiency

D2: Impact on the process user

D3: Business success

D4: Direct operational success

D5: Process improvement fit and preparation for future

Fig. PABRAHAM.1 : Success Dimensions' achievability in relation with SPI project's time frame

The Use of Dimensions in Measuring Success

How can the success of an SPI be assessed using the dimensions identified in this paper? Is a success in SPI a simple of sum its dimensions?

Theoretically, when following Law, Wong and Mobley's [13] taxonomy of multidimensional constructs there are three possibilities for determining the relationship between the dimensions and the construct: 1) latent model, 2) aggregate model and 3) profile model. The first two shall be addressed here.

If 'success' were to belong to the latent model, its dimensions would simply be different forms (of successes) manifested by the construct. Success would therefore be a higher-level construct that underlies its dimensions. When following closely Law et al.'s argumentation on Farmer et al.'s [5] multidimensional construct on upward influence tactics, one could conclude that there exists a general multidimensional construct called 'success in SPI' which underlies five distinct dimensions that are also multidimensional constructs themselves. This line of reasoning could be confirmed when analysing the method by which Shenhar et al. [18] arrived to their success dimensions. Originally Shenhar et al. identified 13 success measures from the project management literature and from the initial phases of their study. Initially these measures were grouped into three dimensions but in their second phase of the study where total number of 127 projects were analysed factor analysis results suggested four distinct underlying dimensions (these dimensions were used as basis for defining success dimensions in SPI project). Therefore in their study, Shenhar et al. defined four success dimensions as first-order factors underlying the 13 measurement items. These four success dimensions are multidimensional constructs themselves where overall success in general is the high-order abstraction.

Latent model classification, however, suggests that any of the first-order factors (e.g. project efficiency) would equally manifest success in SPI. Theoretically speaking this could be possible but in operational use the results would be confusing and misleading since any single dimension diverges the view on the level of success if others are totally left out of the consideration. Therefore, the second model of multidimensional constructs – aggregate model suggested by Law et al. [13] is more appropriate when considering the operationalisation of the concept of success. Aggregate model enables us to determine a formula by which the level of success may be determined. Under the aggregate model classification the 'level of success achieved in SPI' is formed as an algebraic composite of its five dimensions. Weights may be used to stress the importance of certain dimensions over the others. For example, for some stakeholders the level of success in project efficiency may not be as important as achieving the business goals. Therefore, more weight should be placed on the business success dimension than on the project efficiency dimension in the evaluation process. This may prove to be difficult however, since some dimensions are more difficult to measure than others. As shown earlier in the Fig. PABRAHAM.1 all success dimensions cannot be assessed simultaneously. Therefore we label success dimensions as issues that characterise success in SPI. Future research efforts should tackle the measurement of each dimension. Tentatively it is hypothesized that the dimensions have the following (see Table 2) characteristics in terms of measurements.

	Success Dimension	**Type of measurements**	**Relative estimation difficulty**
D1	Project Efficiency	Hard measures (e.g. work effort)	Low
D2	Impact on the Process User	Soft measures (e.g. satisfaction, ease of use; work morale; level of excitement)	High
D3	Business Success	Hard measures (e.g. productivity)	Moderate

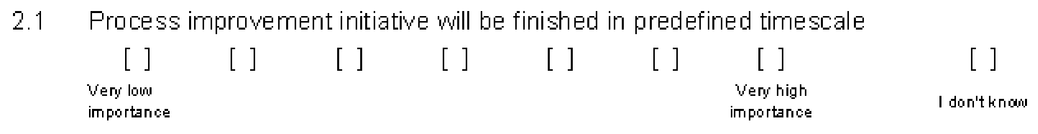
	Success Dimension	Type of measurements	Relative estimation difficulty
D4	Direct Operational Success	Hard measures (e.g. defect ratio)	Moderate
D5	Process Improvement Fit and Preparation for Future	Both (experience database)	High

Table PABRAHAM.2 : Measuring success dimensions

Results From the Empirical Analysis

Importance of Success Dimensions

A total of 23 change agents from 11 different organizations volunteered to evaluate the relative level of importance of each dimension from their viewpoint. Change agents were software developers, designers and process improvement staff who have responsibilities for process improvement activities. A 7-point Likert scale ranging from very low importance to very high importance with 10 items was designed to rate the importance of five success dimensions. A possibility to answer ‘I don’t know’ was given separately apart from the 7-point scale for each item. A sample item of the first dimension (project efficiency) is illustrated below.



2.1 Process improvement initiative will be finished in predefined timescale

[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
Very low importance						Very high importance	I don't know

Fig. PABRAHAM.2 : A sample item from the questionnaire

Early results (Fig. PABRAHAM.3) indicate that change agents value all assessed categories to be at least moderate level of importance in regard to evaluating the success of an SPI initiative. Shenhar et al. [18] found out that project managers place greater importance to the customer (in engineering projects) than they attribute to commercial success or any other project’s impact on the organisation. Similarly in the present study the change agents evaluated the process user satisfaction (dimension 2) having the highest importance above all other items or dimensions. Indeed, the importance of people issues in improving software processes has been acknowledged since only three of 23 (13%) respondents evaluated the process user satisfaction as an indicator of process improvement success lower than high importance (below six in respective scale).

Somewhat contradicting finding with the SPI literature is the result that achieving the business goals together with process improvement initiative’s fit to overall SPI programme and its ability to prepare the organisation for future improvement initiatives were ranked having the lowest importance from change agent’s point of view. This finding provides preliminary support for the idea that any measurements considering success in SPI should incorporate more than one single viewpoint in order to provide an adequate view on process improvement initiative’s success.

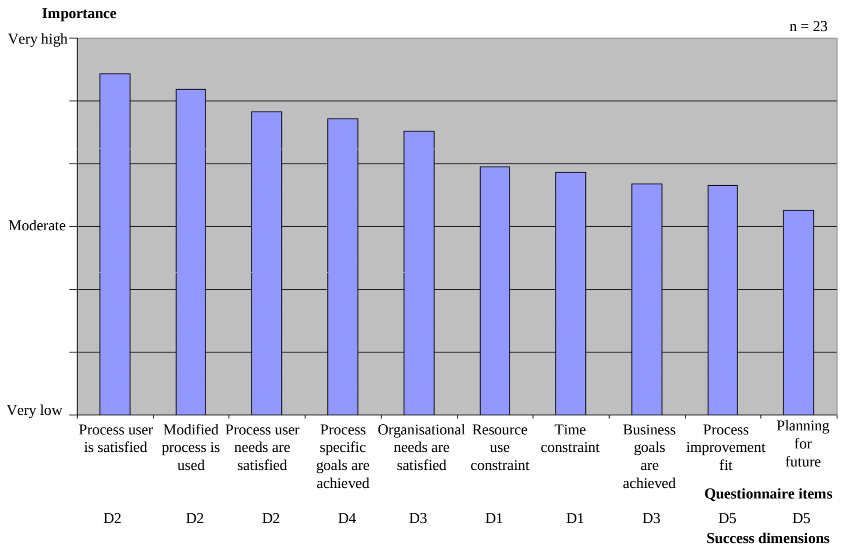


Fig. PABRAHAM.3 : The relative importance of success dimensions in SPI from the change agent's viewpoint

Evaluating Overall Success

Even though, as indicated earlier in the paper, the issue of direct measurement is not the main concern of this paper, for practical reasons[4] the change agents (17 of them) were to evaluate the level of success they achieved from their point of view in recent SPI initiative they were responsible for. In addition changed agents evaluated the overall success they achieved using a 7-point scale.

As suspected, direct business benefits (illustrated below, in Figure 4) were the most difficult to assess since only 41 % (7 of 17) did attempt to give an evaluation of it. The 'easiest' (or the most frequently assessed) one to assess was the project efficiency dimension (D1).

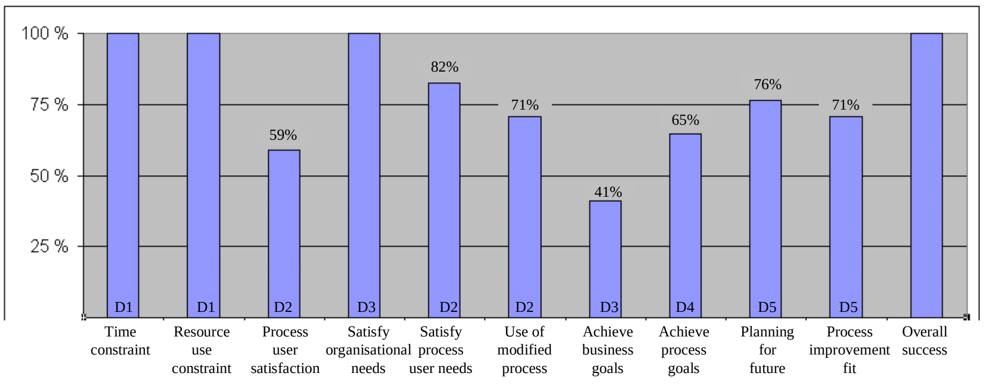


Fig. PABRAHAM.4 : Percentage of respondents per success item

To our surprise, however, there was a remarkably small difference between averaged level of achievement value (calculated as a simple sum) and weighed average (the level of importance was taken into consideration for each item) (see Fig. PABRAHAM.5). Also, the self-evaluated total level of success corresponds rather well to calculated values. Only in three occasions there is a noticeable difference. One should note that the higher the number of items assessed is, the higher should be the reliability[5] of the assessed level of achievement. It is interesting to note that the first four evaluations

[4] It was also studied whether the level of management commitment demonstrated would influence the success level achieved. Results are reported in [2].

[5] The quality of any assessed item depends on the quality of the data that was used as a basis for the evaluation. If all values are just best

were done for the same SPI initiative by four different change agents. While the results (achievement levels) are extremely similar, the number of items used for evaluation varied considerably (from 4 to 10). This calls for attention on the subjectivity of the assessments when there is little data behind the evaluations.

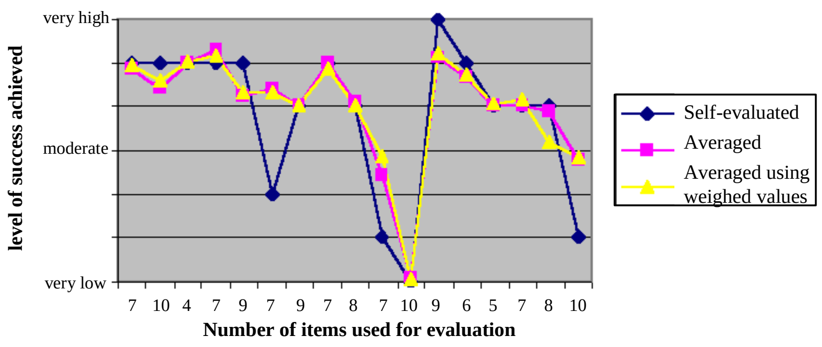


Fig. PABRAHAM.5 : The level of success achieved in SPI

Conclusion

Practice has demonstrated that change agents, quality managers, process owners and researchers are often troubled in defining the level of success achieved in SPI. Analysis showed that existing literature on SPI success has not considered all relevant stakeholders and different levels of successes. To fill this gap five success dimensions that can be used to characterise success in software process improvement initiative were introduced. These dimensions were (1) project efficiency, (2) impact on the process user, (3) business success, (4) direct operational success and (5) process improvement fit. Change agents evaluated the relative level of importance from their point of view and the results indicated that a) all process success dimensions were evaluated to be at least moderately important to change agents, b) the process user satisfaction dimension was rated the highest while c) the ability of an SPI initiative to prepare the organisation for future initiatives was ranked the lowest.

Having the dimensions identified, practitioners may find them useful e.g. in popular GQM method where metrics are derived from agreed goals by formulating targeted questions (see details in [19]). Dimensions point to the direction of the type of questions that should be asked when considering success in an SPI. Future research will concentrate on operationalising and validating suggested success dimensions. This paper reported the importance of each dimension from the change agent point of view. This view will be expanded to include other stakeholder views also. Shenhar et al. [18] proposed that achievability of various success dimensions varies depending on the type of project. Therefore, effort will be expanded on categorizing various types of software process improvement activities into a set of categories reflecting differences in the extent and type of change intended. Also, the complex set of relations between the dimensions should be defined and validated. These research activities should enable us in defining more rigorously measures that explain proposed success dimensions.

We have taken a step toward a mature measurement of success in SPI. Hopefully practitioners find these dimensions useful in explaining the level of success achieved in their software process improvement programmes. Additionally, having introduced the dimensions both researchers and practitioners are able to communicate failures and successes of process improvement initiatives using the same typology. This enables bridging the gap between science and practice and helps unifying lessons learned reports from the industry.

guesses, then the overall achievement level is also ‘only’ a best guess.

Acknowledgments

Author would like to express his gratitude to Dr. Shenhar at Stevens Institute of Technology for providing the material on his study, Dr. Seppänen at University of Oulu, Ms. Parviainen at VTT Electronics and Dr. Kilpi at Nokia Mobile Phones for their insightful comments on the earlier version of this paper.

References

1. Abel-Hamid, T.K. and Madnick, S.E. The elusive silver lining: How we fail to learn from software development failures. *Sloan Management Review*. 39-48.
2. Abrahamsson, P., Is Management Commitment a Necessity After All in Software Process Improvement? in *Proceedings of the 26th Euromicro Conference*, (Maastricht, The Netherlands, 2000), IEEE Computer Society, 246-253.
3. Collier, B., DeMarco, T. and Fearey, P. A defined process for project post mortem review. *IEEE Software*, *13* (4). 65-72.
4. El Emam, K., Goldenson, D., McCurley, J. and Herbsleb, J. Success or Failure? Modeling the Likelihood of Software Process Improvement, International Software Engineering Research Network, 1998.
5. Farmer, S.M., Maslyn, J.M., Fedor, D.B. and Goodman, J.S. Putting upward influence strategies in context. *Journal of Organizational Behavior*, *18*. 17-42.
6. Fitzgerald, B. and O'Kane, T. A longitudinal study of software process improvement. *IEEE Software*, *16* (3). 37-45.
7. Freeman, M. and Beale, P. Measuring project success. *Project Management Journal*, *23* (1). 8-17.
8. Grady, R.B. *Successful software process improvement*. Prentice Hall PTR, Upper Saddle River, NJ, 1997.
9. Herbsleb, J.D. and Grinter, R.E., Conceptual simplicity meets organizational complexity: case study of a corporate metrics program. in *Proceedings of the 1998 International Conference on Software Engineering*, (1998), 271-280.
10. Humphrey, W.S. *Managing the Software Process*. Addison-Wesley Publishing Company, Inc., 1989.
11. Kasse, T. and McQuaid, P.A. Entry Strategies into the process improvement initiative. *Software Process - Improvement and Practice*, *4*. 73-88.
12. Kitchenham, B.A. *Software metrics: measurement for software process improvement*. Blackwell Publishers, Cambridge, Mass., 1996.
13. Law, K.S., Wong, C.-S. and Mobley, W.H. Toward a taxonomy of multidimensional constructs. *Academy of Management Review*, *23* (4). 741-755.
14. Millet, D. and Powell, P., Critical success factors in expert system development a case study. in *Proceedings of the 1996 conference on ACM SIGCPR/SGMIS Conference*, (Denver, CO, 1996), ACM, 214-222.
15. Nolan, A.J. Learning from Success. *IEEE Software*, *16* (1). 97-105.
16. Pfleeger, S.L., Jeffery, R., Curtis, B. and Kitchenham, B. Status report on software measurement. *IEEE Software*, *14* (2). 33-43.
17. Reo, D.A., Quintano, N. and Buglione, L., Measuring software process improvement: There's more to it than just measuring processes! in *2nd European Software Measurement Conference - FESMA'99*, (Amsterdam, the Netherlands, 1999).
18. Shenhar, A.J., Dvir, D. and Levy, O. Project Success: A Multidimensional, Strategic Concept, Stevens Institute of Technology, Hoboken, NJ, 1999.
19. van Solingen, R. and Berghout, E. *The Goal/Question/Metric Method: A Practical Guide for Quality*

Improvement of Software Development. McGraw-Hill, Cambridge, UK, 1999.

20. Xia, F., On the danger of developing measures without clarifying concepts. in *Proceedings of the Asia Pacific Software Engineering Conference*, (1998), IEEE Electronic Library online, 86-92.

21. Zahran, S. Software process improvement: practical guidelines for business success. Addison-Wesley Pub. Co., Reading, Mass., 1998.

About the author

Pekka Abrahamsson is currently working as a research scientist at the Department of Information Processing Science in the University of Oulu in Finland. He holds Master's degree in Information Processing Science. In the beginning of 1999 he was accepted to the Infotech Oulu Graduate School, to prepare his doctoral thesis as a full-time researcher. His research interests focus on exploring the concept of commitment in software process improvement initiatives. The study aims at creating an adaptable commitment process for software development organizations. His professional experience involves three years in software development and three years in quality management, where he was responsible for the process improvement programs.

Now you see it …
… Now you don't

Tim Hind
AXA Sun Life Services, Bristol, UK

Introduction

AXA is one of the largest insurance groups in the world. It has over 140,000 people (employees and agents) in over 60 countries. There are over 60,000 in Europe. Assets under management at the end of 1999 were € 781 billion. [1]

When Sun Life and AXA Equity & Law merged in 1997 the UK market had a new major player. AXA Sun Life has become the 4th largest UK Life Company. Over the 4 years it grew from 3,300 staff to 4,700. IS staff increased from 500 to 750 in the same period. They were supported by 3 metrics staff throughout that time. [2]

Over the same 4-year period the systems base has changed dramatically. Whereas Sun Life used to be predominantly IBM Mainframe based with little or no cross-site working the office had started to migrate all of its systems to Tandem about 2 years ago. Cross-site working has been a significant feature of the last 2 to 3 years. Recently migration to Tandem was halted and a reversion to MVS started.

Likewise, the PC environment has shifted from being a multiplicity of operating systems and versions of software to being a standard Windows NT build for all staff.
The use of statistics to monitor IS activity has been in place for over 20 years. During that period the focus has changed regularly as the office has moved from just looking at Cost Apportionment to measuring IS Productivity through internal and external benchmarking.

Providing a way of demonstrating productivity had become an important part of Sun Life's process from early in 1996. This paper looks at the challenges associated with the establishment of a Metrics Programme to track productivity over the 5 years of activity 1996 – 2000. The visibility of the metrics deployed plays an enormous part in the acceptance of any Process Improvement activity. It is my conjecture that visibility is vital to enabling behavioural change. When, for whatever reason, the metrics cease to be visible it becomes more difficult to challenge (and hence change) behaviour.

Metrics in the Sun

In order to justify the claim that visibility is vital to enabling behavioural change, I am going to illustrate the initiatives that we have engaged with over 20 years.

The story begins with a strong accountancy focus and the need to be able to apportion mainframe costs between different users. It builds towards a model for process improvement and the metrics required to deal with quality, delivery & cost of development projects.

In many respects there is a '3 steps forward – 2 steps back' feel to each of the initiatives. In each case, the progress made following the start of the initiative is evident. In each case, the relevance of the metrics is linked to the visibility of the results.

Metrics in the Sun – Apportionment

Metrics started in Sun Life in about 1980 and was specifically aimed at helping the bean-counters (sorry - the cost accountants) to split project costs out according to the beneficiary of the product. It was done in several different ways over the ensuing years as the process became more sophisticated. Some of the different ways are outlined below: -

Equal Division

Each corporate department that was using resources bore its equal share of the costs associated with production running and / or development.

This is very unsophisticated but can be used

- when there is no reason to do otherwise – e.g. where the systems of each department are uniform in complexity
- where there is no other mechanism to use – e.g. where the organisation is not yet mature enough either to know to do otherwise or to have the ability to measure differently.

Pro Rata by Time

Each department receives a charge based on the Computer / Personnel Time (as appropriate) spent on the activity being charged.

This works well where the time spent is reasonably proportional to the impact on cost. Where it began to fall down for us was when the administration systems were compared against the actuarial systems. In this case the highest costs, at the time, fell on the purchase of memory devices. Administration used a lot of storage, whereas Valuation used an enormous amount of core processing time. The actuaries were aggrieved.

Pro Rata by Storage

An alternative here is where the division is done according to the amount of storage used by the systems. See previous paragraph to see where this breaks down.

Pro Rata by Activity

For this to be done well there needs to be analysis of the activities which take place and for a value to be placed on each of them.

Subsequent analysis of which corporate divisions use which activities creates a framework for apportionment.

The sorts of activities to be used include: -

- core processing – uses time as a measure
- print a sheet – uses count and whether duplex or simplex
- access a database – uses time (which compensates for any complexity inherent in the tables)

Pro Rata by Product

The highest level of analysis which uses the outcome from activity based costing (above) and applies the intelligence associated with '*what sort of*' and '*how many of*' the activities go to make up the process of making a given product.

This amalgamation of systems and marketing information works by estimating usage at the beginning of a period and applying the resultant knowledge to obtain a standard charge for each product. These products may include: -

- do a quotation
- print a policy
- surrender a policy

The following is a brief description of the process which we went through to achieve this :-

- Split Costs between Development & Operational functions
- Split Operational Function Costs between Development & Production.
- Split Operational Function Costs by area supported – CPU, Storage & Printing
- Analyse Usage of CPU, Storage & Printing to enable identification of Unit Costs
- Analyse System/Application Usage of these to apportion costs to Systems.
- Analyse Business Functions and Volumes.
- Analyse System/Application Functionality and Map to Business Functions.
- Apply Costs to Business Functions to derive unit Business Function Cost.

To illustrate this, our CP Second cost (*Analyse Total Usage of CPU, Storage & Print Facilities to enable identification of Unit Costs*) was £0.606 in 1993 based on a forecast usage of 147m CP seconds at a cost of £8.9m. Likewise Storage came out at £0.2622 per Gigabyte Hour.

The total budget for Production was around £6m of which £2m related to our Unit Linked Pension System. Out of this about £191k was required to support 32k New Business Transactions leading to a cost per unit of New Business of £6.

Having set the rates across all business functions and contract types the costs were recovered by reference to Business Volumes and an adjustment made at the end of the year to compensate for any variation in assumptions.

Conclusion

It should be noted that these bases for the subdivision of costs were followed almost chronologically. It is possible to see the evolutionary process through them. Each was followed until its visibility brought it into disrepute.

Because many were used to impact the bottom line for some subsidiary companies they were highly visible. When circumstances changed and the need for greater sophistication in measurement arose –

so the metric process changed to reflect that need.

Metrics in the Sun – Productivity

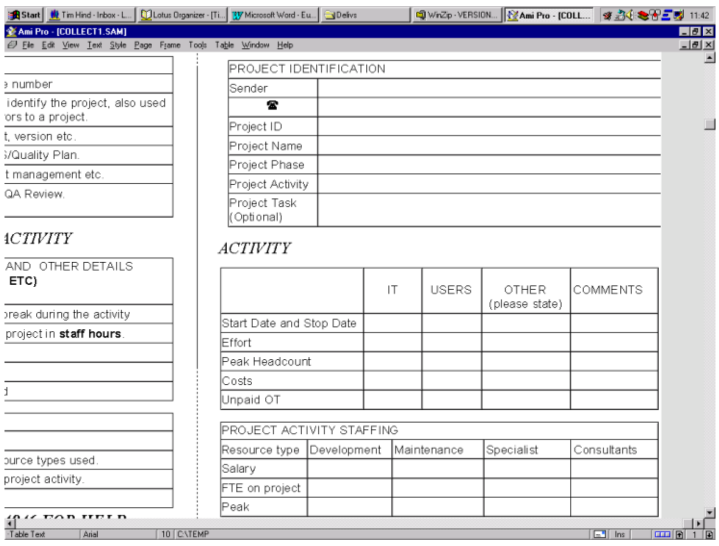
The year 1996 saw the introduction of a formal Metrics Programme to concentrate on the issue of productivity within the IS Division. All of the above activity was still going on in the background but was being dealt with in a pragmatic way by carrying out exercises from time to time to confirm the accuracy, or otherwise, of the parameters being used for the apportionment.

Our senior IS Management were being continually told that their IS Developments were inefficient. To answer that criticism it was decided to create a Metrics Team whose first activity was to organise an External benchmark to measure Productivity. Our intention was to deal with Development in the first year and to follow quickly with Maintenance, Data Centre and Human Resources.

Our first year of operation soon helped us to realise that we were being over-ambitious and so we collapsed the operation to look at Development only. We retrospectively captured data from the previous year's projects. We were very popular as you can imagine! We carried out the external benchmark using CSC Index. The results were briefed to the Senior Management Team.

The process of benchmarking revealed us to between the upper & lower quartiles in a survey of 60 companies in 4 out 5 of the measures used by CSC. The one where we were below the lower quartile (48th ranking out of 60) was the measure that we had chosen as our productivity measure (Function Points / Staff Month). This was due to the presence of 3 large poorly performing projects. We consoled ourselves that we were, nevertheless, cheap, fast and reasonably reliable operationally.

We rapidly understood that a Data Collection Book was necessary for any subsequent benchmarking activity and so a 56-page word based Data Collection Book was created. An example of one of the pages from this Book is shown below.
This page was used to capture information at Activity level. Other pages existed for Project Level & Task related data.



e number
identify the project, also used ors to a project.
t, version etc.
S/Quality Plan.
t management etc.
QA Review.

ACTIVITY

AND OTHER DETAILS ETC)
oreak during the activity
project in **staff hours**.
ource types used.
project activity.

PROJECT IDENTIFICATION	
Sender	
☎	
Project ID	
Project Name	
Project Phase	
Project Activity	
Project Task (Optional)	

ACTIVITY

	IT	USERS	OTHER (please state)	COMMENTS
Start Date and Stop Date				
Effort				
Peak Headcount				
Costs				
Unpaid OT				

PROJECT ACTIVITY STAFFING				
Resource type	Development	Maintenance	Specialist	Consultants
Salary				
FTE on project				
Peak				

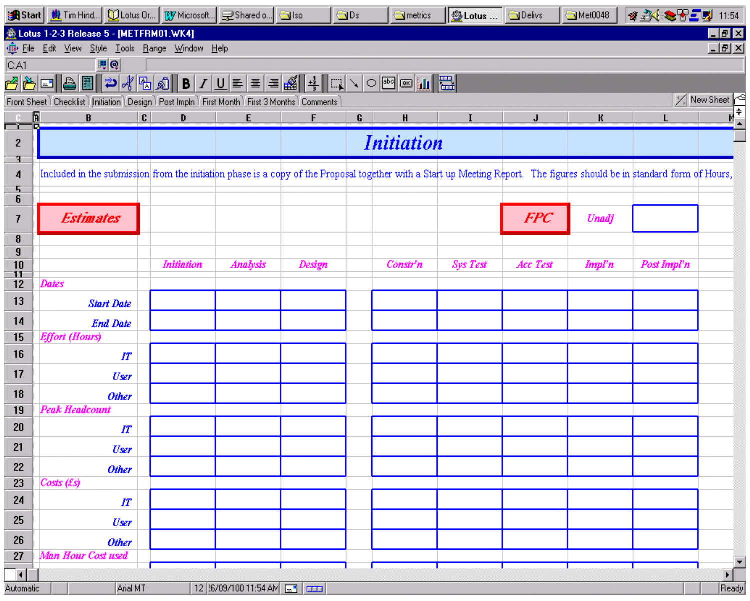
Fig. TCH.1 Activity Sheet from 56-page DCB

Although this addressed the criticism about retrospective data collection it caused problems of its own. As a document it was bulky. It had been quickly produced and poorly implemented and so misunderstandings arose about its usage.

Rolling out any new thing requires considerable effort and planning, involving presentations and roadshows. Even the language needs rehearsal to ensure that everyone understands the same thing from a particular word. As Kim Caputo says in her book on the CMM when talking about implementing improvement efforts without planning 'The managers' reaction was something this : … „I'm not going to interrupt their work for this. I don't see the benefit in doing this."'. [3]

However, the book's existence meant that the following year's benchmark was a much smoother operation. We raised our Productivity Position to 42nd and were now above the lower quartile. In fact the inclusion of an particular project had placed us 25th but we felt that its removal was more representative of our place. Even being in 42nd had raised our productivity by nearly 50%.

Project Managers were even happier once a revised 8 page Data Collection Book was launched. A more attractive style (using Lotus 123) was employed. The use of colour enabled more information to be clearly asked for. The page illustrated was collected at Initiation and gave details of the Estimated Cost, Resourcing and Effort profiles.



Initiation

Included in the submission from the initiation phase is a copy of the Proposal together with a Start up Meeting Report. The figures should be in standard form of Hours,

Estimates

FPC Unadj

	Initiation	Analysis	Design	Constr'n	Sys Test	Acc Test	Impl'n	Post Impl'n
Dates								
Start Date								
End Date								
Effort (Hours)								
IT								
User								
Other								
Peak Headcount								
IT								
User								
Other								
Costs (£s)								
IT								
User								
Other								
Man Hour Cost used								

Fig. TCH.2 Activity Sheet from 8-page DCB

Interestingly, although the Project Managers were offered this new book electronically, they opted to complete it on paper.

Metrics in the Sun – Merger

During that year, 1997, Sun Life began the process of merging with AXA Equity & Law.

The process of attempting to harmonise merged IS activities was fraught with opportunities. We carried out a gap analysis of different processes. We retrospectively captured the data for the previous year's AXA Equity & Law Projects. Our popularity rating was now at rock bottom and digging.

The subsequent joint benchmark was interesting as there followed almost endless discussion of the perceived similarities and differences.

One difference in particular that caused us grief was the productivity measure itself. We had adopted Function Points / Staff Month as our principal measure. The two companies had a different view of how many days constituted a Staff Month – one used the basis of 207 days in a year (17.3 productive days per month) and the other company was looking at 175 per annum (14.6 per month). The essential differences lay in what activities were included in the productive month (team meetings, training etc).

Another issue arose over the inclusion or exclusion of the perceived added value from architectured reuse of code. This was defused by taking two different measures – effectiveness (which allowed for reuse) and efficiency (which was looking purely at current development activity).

The merger put a stop to a lot of our activities. This was partly due to the increasing challenge to 'old orders' from both sides to the merger – each side took it as an opportunity to move away from the established ways of performing. Part was due to the difficulties of identifying what work was going on while the world was 'unharmonised'.

These 3 years produced a lot of valuable information. We set ourselves targets to improve and have had a measure of success as illustrated by the early part of the graph shown.

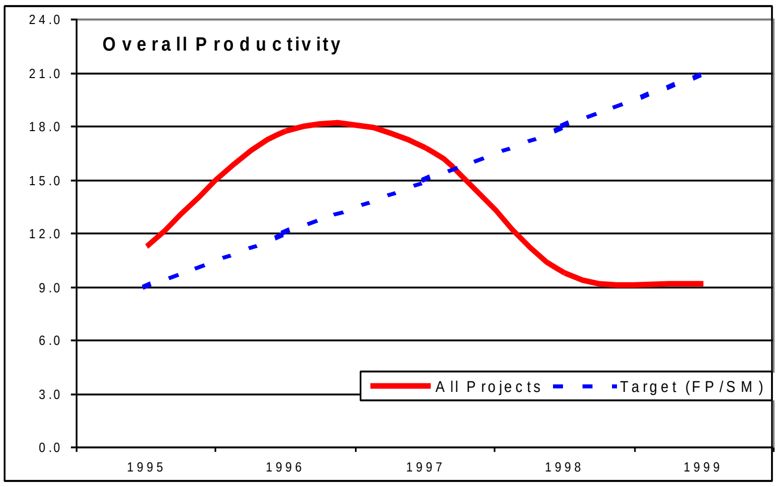


Fig. TCH.3 Overall Productivity - Graph

The easiest mistake to make is to assume that the rest of the graph shows a step backwards. In fact there is still some good news to be gleaned from this by further investigation.

With the merger came the pressure to do small to medium enhancements and as a result all the major (very productive) green field & package developments disappeared from our portfolio. There was also the fact that we had to make changes to two different systems at two separate locations.
Analysing medium-sized enhancements separately shows a different picture.

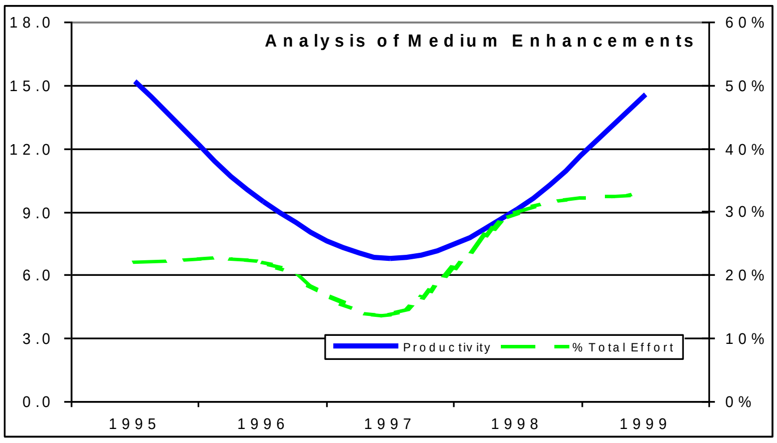


Fig. TCH.4 Medium-Sized Enhancements - Graph

Here it can be seen that the level of Medium-Sized Enhancements within the portfolio changes over time. By 1998 the level was at its highest and at least double the value for 1997. In what was clearly our core activity we had retrieved our previous productivity position and were showing strong improvement as the table below shows.

Productivity	**1995**	**1996**	**1997**	**1998**	**1999**
No of Projects included	20	28	43	18	23
Target (FP/SM)	**9.0**	**12.0**	**15.0**	**18.0**	**21.0**
All Projects	11.2	17.8	16.8	9.8	9.2
All Enhancements	11.4	8.3	8.6	5.9	9.7
Major (>1000 FP)	10.9	4.5	9.5	n/a	n/a
Medium	**15.2**	**9.5**	**6.8**	**9.1**	**14.6**
Minor (<250 FP)	5.3	11.2	10.7	2.8	7.0

Fig. TCH.5 Productivity – Table

Metrics in the Sun – Performance Improvement

Having spent several years sharpening our data gathering tools we had not really progressed very far. And yet we were arguably ahead of the game. The adventure continues with the move towards a more positive Performance Improvement culture.

You will have noticed that I used the phrase ‘Performance Improvement’ rather than ‘Process Improvement’. This was due to the need to focus on what improved performance, which may not always be due to a change of process. For example, it is possible to improve performance by use of tools or increased functionality of tools used.

We started a programme designed to change our Development activity. There were workstreams catering for Process, Tools, Skills and Metrics. The programme took on the title of PRIDE which, apart from being one of AXA’s core values, was able to be carved out of the words ‘**P**e**R**formance **I**mprovement in **DE**velopment’.

Normally, when naming a project or programme the name is of little consequence. It may act as a focus but little else. In this instance, however, it seemed so appropriate that we should use the word PRIDE as it would only be through pride in oneself, one’s work and in the company that we would be able to obtain sustained improvement. It would be necessary, in our opinion, to change the culture of the company to achieve the goal of the programme.

The four workstreams were interrelated. It was important to improve the processes and the people capabilities and so the Process & Skills phases were designed to do that by concentrating on SEI CMM and the IiP (Investors in People) standards. To support the people it was going to be essential to implement better support through Tools – in particular the introduction of a universal time-recording & project management system. The Metrics workstream was there to provide evidence of the journey towards a more mature culture.

Before embarking on our Programme we had carried out an informal mini-assessment against the SEI Capability Maturity Model and obtained recognition of our place on the scale. I will leave you to complete the sentence ‘AXA Sun Life are CMM Level …’

At the same time as this was going on we were going through our second (or was it third) merger!

We had it all planned and then were frustrated by a massive restructuring exercise which put our plans on hold. Our plan had been to achieve Level 4 by the end of 2002. We were held up for at least 9 months.

During the period of uncertainty, the thing that we did know was that there would have to be improvement in processes. So although the decision about whether or not to go for CMM was uncertain, we made progress in our reinvention of Delivery Lifecycles and implemented ABT as our time-recording mechanism and project repository. With this new tool there came a resistance to the collection of other data. Our metrics data collection process came to a halt.

Concurrently, the restructuring process itself enabled us to rethink many of our relationships within the business and new Governance processes and Management Reporting tools emerged. Strangely enough, many of the things that we would have achieved with our PRIDE Programme came out of the turmoil – almost by osmosis.

Next year our formal assessment against the SEI CMM was to have taken place. However, with the new strategy emerging it looks like we will be outsourcing much, if not all, of our Development activity and so the balance of our work will probably be better assessed against the CMMI. I am convinced that we will be in a much stronger position. Watch this space!

Not all of this has happened by having a cast iron plan in place. Sometimes, just sometimes, a little bit of luck, serendipity, is involved.

Visible Management

You will have noticed that there has been a learning process going on over the five years of operation. We had started with a simple question and answer required - relating to the efficiency of our operation. We moved towards a much more sophisticated set of questions and answers came more by luck, initially, than by good judgement.

At the start of the process we had used some simple visible management techniques. We had a simple question and a simple illustration was a suitable response. In the example in Fig TCH.6 we included each project for the year together with its productivity rating. For ease of use they were sorted by the value of the measure.

This was one of the earliest versions of this graph. The result of the presentation of information in this form was a large activity amongst the seemingly low productivity projects to attempt to change their position in the ‘table’. One of the problems of simplicity is that the subtle differentiators between projects get ignored.

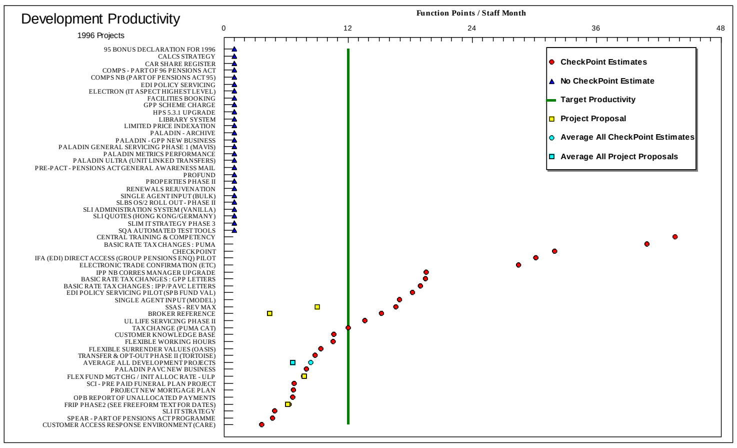


Fig. TCH.6 Productivity – Graph

At that time, we had total commitment from Senior Management and a 3-line whip on completion of the task. We published the results, Middle Management responded and things began to change. In particular, one change resulted in the classification of projects by language, platform etc. This enabled a more sophisticated set of results to emerge which reflected this categorisation.

So long as these two things occurred – i.e. the question remained simple and the Senior Management remained committed we were on to a winner.

None of our current Senior Management team has continuous service over the five years. As soon as it began to change, priorities changed and the Middle Management used it as an excuse to duck out of the process of delivering metrics data.

When the question changes you have to use different techniques to process the information. The simple charts that we used to show that we were becoming more or less productive gave way to a more complex set of graphs to attempt to show how the process was improving – we called this **Project Profiling**.

At the same time as we began to deal with Project Profiling we were being challenged by our French parent company to look at the use of a simple **Scorecard** with which to assess Corporate Maturity.

Above all, whatever techniques or tools are to be used, the commitment of Senior Management is essential to the success of the metrics enterprise.

The next few sections show a little of the joys and sorrows of the use of these techniques

Project Profiling

For years we put away the data collected and nothing appeared to come out. This lack of visibility creates major difficulties. It is said that most Metrics Programmes last less than 18 months. This is not surprising as the average attention span of a CEO is less than 6 months and it takes a year to plot two significant points on a graph.

Project Profiling is a way of making historic data visible for improving the estimating process and monitoring progress. We started to look at this technique when we were 3 to 4 years in to our Metrics Programme.

The technique requires enough information to be available before you start. It is no use analysing the data at the lowest level and hoping to be able to draw any useful conclusions consistently. It is necessary, therefore, to start at enterprise level and only subdivide when there is sufficient data within the subdivision to enable a statistically viable conclusion to be drawn.

It needs sustainable process to ensure that the data is maintained and then improved. This presupposes the presence of serious management commitment.

What we have done is to take the effort data across the lifecycle and mapped it by phase. Using this data we have shown that our average project spends the following percentage of time on key areas: -

Initiation	Analysis	Design	Construct'n	System & Integration	User Acceptance	Implement'n	Post Implement'n
6%	8%	14%	29%	27%	9%	4%	3%

Fig. TCH.7 Project Profiling at Enterprise Level

This is an enterprise view, which would need careful consideration before being applied universally or even to a specific project. To do so, one would need to be able to answer questions about how close the average project was to the one under investigation. Is it the same size? Are the language and platform similar? Are the skills of the staff comparable? And a host of other similar questions come to mind.

Looking specifically at the above table and analysing it by category it is clear that wild variations do occur. In particular the number of package implementations has a dramatic impact on the Construction % for 'Other' projects.

	Initiation, Analysis & Design	Construction	Testing	Implementation & Post Implementation
All Projects	28%	29%	36%	7%
Enhancements	27%	26%	41%	6%
New Build	28%	33%	34%	5%
Other	33%	26%	30%	11%

Fig. TCH.8 Project Profiling by Category

When we attempted to analyse this by year of operation we discovered some useful insights. Fig TCH.7 split by the 5 years of operation 1995 – 1999 are given in Fig TCH.9 below.

This shows evidence that the way in which we have performed over the five years has changed. The time spent on Initiation, Analysis & Design has risen from 30% in 1995 to 38% in 1999 with a significant impact on Testing which reduced from 35% to 25%.

Year	Initiation	Analysis	Design	Construct'n	System & Integration Testing	User Acceptance Testing	Impl'n	Post Impl'n
1995	6%	8%	16%	27%	26%	9%	4%	3%
1996	6%	5%	14%	24%	34%	10%	6%	2%
1997	5%	9%	14%	31%	27%	9%	3%	2%
1998	8%	8%	13%	30%	20%	9%	8%	4%
1999	9%	11%	18%	29%	22%	3%	3%	5%

Fig. TCH.9 Project Profiling at Enterprise Level – Split by year of operation

We realise that we are only looking at this relatively, but it is still a major shift in operational terms over a comparatively short time. We have incorporated this information into our Best Practice Manual and it has been used as a basis for our estimating process within ABT.

Project Profiling can be used to monitor progress. One example will help to illustrate this. A project was invited to publish its estimates which we plotted against our standard 'Enterprise Level' profile. *Names changed to protect the innocent!*

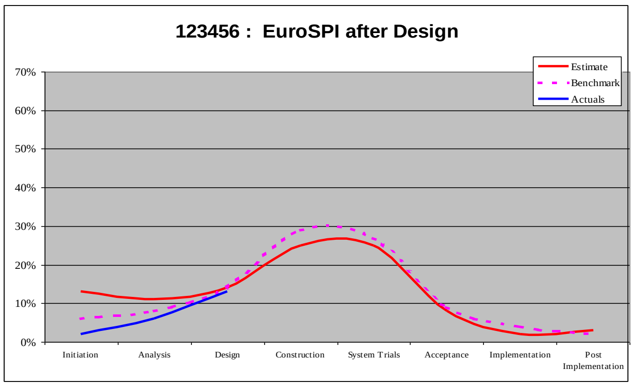


Fig. TCH.10 Project Profiling – Project at the end of design

It was evident that the Estimate indicated a higher percentage of effort being planned for the initial stages – but the actuals were much lower. They were even lower than the Enterprise expectation.

It is not surprising, therefore, that when the final data for the completed project was in that the project spent a lot more on construction due to poor analysis & design preparation. This was our first attempt at predictive analysis.

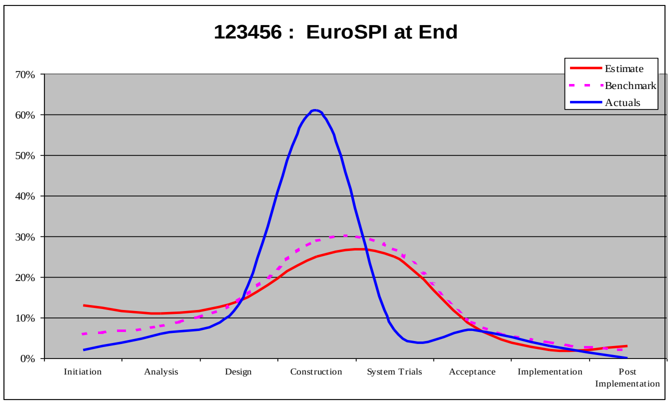


Fig. TCH.11 Project Profiling – Project at Post Implementation

So the accumulated data has been used to provide a starting point for estimating and then also used for comparison with actuals. What can be said is that Project Profiling seeds a process, which can then be improved as more projects enter their data into the repository.

Scorecards

Balanced Scorecards came to prominence thanks to Robert Kaplan and David Norton [4]. The idea came from a desire to convert Strategy into measurable objectives. It consists of Financial, Customer, Process and Learning & Growth aspects and attempts to provide perspective that is often lacking when viewing a single measure (say Return on Investment {ROI}) in isolation.

In our environment this initiative came from our French parent company as a Group-wide initiative at the same time as an internal (to AXA Sun Life) move occurred to identify some key performance indicators.

At Group Level the scorecard was piloted in 1999 across 6 or 7 out of the 60+ companies and fed back through Regional Technology Officers. This year saw the introduction of the scorecard throughout the whole group. This particular initiative has still got a long way to go globally before it will be of any use.

The way the scorecard works is to identify 4 major areas where improvement is deemed to be necessary for AXA to become world class. Within each area there are three themes for which a score (on a scale from 1 to 10) is needed.

Where the scorecard works well is in the spread of themes that are included – Financial Control, IT Capabilities, Innovation & Governance. Where it is not currently working well is in the granularity of the scoring mechanism.

For many of the measures we have included several sub-measures which are then averaged to produce a single value. This means that when the measure changes value it is not possible to identify what it is that has caused the change. Similarly, when a value doesn't change, there is no guarantee that all of the sub-measures aren't leaping around all over the place – but being averaged out to no overall difference.

Locally we had started to look at providing a set of measures based on Capers Jones [5]. The PRIDE programme was to have delivered this but with its suspension pending structural reorganisation we have not yet got beyond the data & measure definition stage.

This local scorecard was aimed at satisfying the goals of improving Productivity, Quality, Skills & Customer Service. In each case the Goal led to one or more measures (14 measures altogether). We had gone through a standard Goal-Question-Metric analysis to get to this point.

There is a recurrent theme through both the Profiling and the Scorecard episodes. They both require – sufficient data, managed procedures, no rocket science and solid Management commitment.

Conclusions & The Future

So the question is 'Where to now?'.

We have learnt an enormous amount through trying things out and by realising what has been successful and what has not. Our future is exciting. I believe it is more mature to have learned from the past and to be in a position to ditch what doesn't work than to hold on to outmoded measures well beyond their sell-by-date.

We have started a new department whose responsibility is to promote Performance Improvement. The visible management bit of this unit sits within a Performance Monitoring Function and meshes neatly with the Software Quality Assurance (SQA) function that I manage.

In SQA there are three basic elements, viz. Process Assurance, Product Assurance & Visible Management

Visible Management here consists of the management of the SQA process itself. In my unit it covers not only the visible management of SQA but also of the Delivery Lifecycle which is under scrutiny.

What we are definitely doing is going back to basics. Having realised that we were trying to be too sophisticated by attempting to apply statistical processes in a culture which, frankly, was not ready to accept such techniques, we are going back to capturing information at a much more basic level.

I may be wrong, but milestone conformance is likely to be as sophisticated as we get in the short term.

So the next question is 'Doesn't this feel like a step backwards?'.

It looks on the surface to be disappointing. But together with the simpler data capture come other goals. In particular – to capture 100% data at this simpler level.

I look forward to being able to show, in a few years time, that this approach delivers a more stable, secure set of statistics over the long term. That is, of course, on the assumption that we have some stable Management commitment to our approach.

Having said that we will not be allowing all the previous work to go by the board. We have already started to do some predictive analysis based on the ABT plans and actuals. With this technology it is possible to take early results from a project and feed back to the projects the possible outcomes. Initial reactions have been positive although the results themselves have highlighted the poverty of the data more than anything else.

At least we can say that we have genuinely started on a continuous improvement process – which must mean we can now aspire to Level 5!

References

[1] AXA Website, www.axa.com

[2] Holt J R, 'Software Metrics - Real World Experiences' in European SEPG 1997 proceedings

[3] Caputo K, CMM Implementation Guide, Addison Wesley

[4] Kaplan R S & Norton D P, 'The Balanced Scorecard', HBS Press

[5] Jones C, 'What it Means to be „BEST IN CLASS“ for Software', November 2, 1998

AXA Sun Life Services

Life & Pensions, Bristol, UK

AXA Sun Life Services was created in 1997 by the merger between Sun Life and AXA Equity & Law. It has become the 4th largest UK Life Company. There was a significant systems impact as a result of this merger.
Sun Life had been created in 1810 when it was separated from its sister company Sun Fire (which continues as part of Royal & SunAlliance). Equity & Law was formed in 1844. Both companies had become part of the AXA group by virtue of Merger & Acquisition activity.
In 1999, the company acquired Guardian Royal Exchange (including its PPP Healthcare wing). In system terms this had a smaller impact than the prior merger.
AXA Group, worldwide, is in the process of rationalising its holding within each country so that branding can be clarified, i.e. it is not confused by the structure of the companies. It has an aim of being in the top 3 in each of its chosen markets in each geographical area of involvement.
The company employs 140,000 people across the globe and more than 60,000 in Europe. Assets under management at the end of 1999 were €781 billion.

Tim Hind

AXA Sun Life Services, Bristol, UK

Born 1950 and educated at Watford Boys Grammar School and St John’s College Cambridge where he studied Mathematics – in particular Fluid Dynamics.

Worked briefly for Royal London Mutual in the Actuarial Department but on leaving University joined Sun Life in London as a Pensions Administrator. In the mid 70s he moved into Data Processing as a Systems Analyst and became a Project Leader in 1980.

His formative years in IS were spent as an analyst on Pensions systems but he has dabbled in almost every other system within the company.

Moved into project statistics in an attempt to assist the cost accountants to split IS costs to business units. After different spells looking at IS Security and working as a Budget Analyst, he joined the IS Metrics Team at its inception in 1996.

Currently managing a team of 3 responsible for Performance Monitoring within IS.

Spare time is taken up with family and church activities as a member of the General Synod of the Church of England.

PRACTICAL MEASUREMENTS FOR REENGINEERING THE SOFTWARE TESTING PROCESS

Stelios PANTELOPOULOS, SINGULAR S.A.
29, Alexandras Ave., 114 73, Athens, Greece
tel. +30 1 647 9600, fax +30 1 6479635,
e-mail: spantel@singular.gr

Yannis PANOPOULOS, *SINGULAR S.A.*
29, Alexandras Ave., 114 73, Athens, Greece
tel. +30 1 647 9600, fax +30 1 6479635,
e-mail: ipanop@singular.gr

Abstract

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

The identification and removal of software defects constitutes the basis of the software testing process, a fact that inevitably places increased emphasis on defect related software measurements. Defect Distribution, Defect Density and Defect Type metrics allow the quantification of the quality on software modules, while Defect Age, Defect Detection Rates and Defect Response Time metrics allow

for pinpointing software inspection and testing process shortcomings. Code coverage and testing effort measurements complement the defect metrics and provide additional software product as well as process, quality indicators. The paper concludes with the presentation of the application of testing metrics in industry focusing on SINGULAR's ESSI STAMP process improvement experiment.

Keywords: Defects, Software Testing, Metrics, Software Improvement.

1 SOFTWARE PROCESS AND PRODUCT IMPROVEMENT

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

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

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

Figure 1 - Customer Satisfaction Matrix

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

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

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

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

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

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

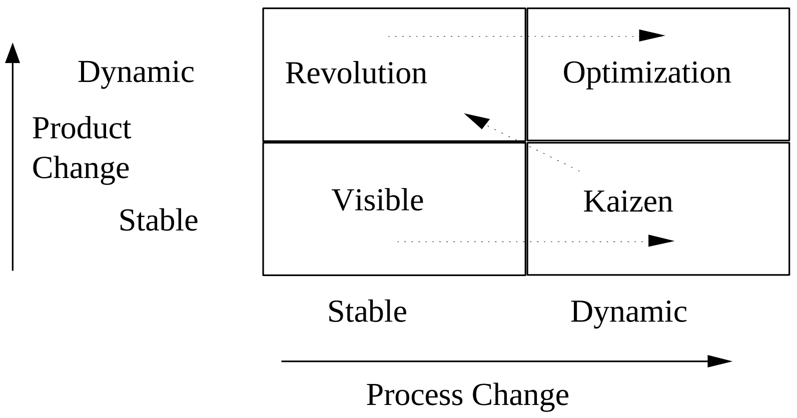


Figure 2 - Process Maturity Navigation Matrix

2 SOFTWARE TESTING PROCESS REENGINEERING

The continuous improvement of the software development process commences with handling it’s weakest link, the testing process. If one accepts that the strength of a chain is equal to the strength of it’s weakest link, the importance of reengineering on the testing process is evident. Prior to reengineering the software testing process, the testing objectives must be established so that the software testing process is in the position to correspond to clearly defined goals. Hetzel [2] lists the following list of practitioner objectives regarding software testing.

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

In order to meet such testing objectives, most software production units that have reengineered their testing process have more or less in some STEP-like testing process. The STEP testing process can be summarized by the following major testing activities [2] :

PLANNING
- PLAN the general approach
- DETERMINE testing objectives
- REFINE the general plan

ACQUISITION
- DESIGN the tests
- IMPLEMENT the tests

MEASUREMENT
- EXECUTE the tests
- CHECK termination
- EVALUATE results

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

3 PRACTICAL TESTING MEASUREMENTS

The software testing process requires practical measurements for the quantification of all software testing phases. Starting from the planning and acquisition phases of the software testing process, one first has to recognize that current software development practices do not segregate coding effort from unit testing, but rather, coding and unit testing are seen as one software life-cycle phase. Therefore, the duration of the testing phase basically covers test planning effort, integration and system testing. The following measurements apply to software development effort distribution [3] :

• Analysis	16%
• Design	17%
• Code/Unit Test	34%
• System/Integration Test	18%
• Documentation	8%
• Implementation/Install	7%

Regarding software development distribution effort, Grady [4] reports the following breakdown from the analysis of 125 Hewlett-Packard projects :

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

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

The project manager must also be in a position to predict the number of test cases required in order to test adequately the developed software. Capers Jones [5] has done extensive research for such measurements and has concluded that the relationship which governs test cases and function points is the following:

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

Capers Jones [6] has also quantified the size of the test plan in pages per function point. The average number of pages created per function point for software project is 0,25 for system software, 0,10 for MIS software, 0,55 for military software and 0,25 of commercial software.

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

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

Pareto	20% of the code causes 80% of the problems 80% of the transactions traverse 20% of the code
Code Coverage	The reliability of software is dependent on the traversed tested code

In order to apply the pareto principle, project data must include details pertaining to the detected defects. Invariably, an analysis of the detected defects allows of a good understanding on the information system strong and weak links. Defect related measurements include the following [7], [8]:

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

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

Table 1: Usage of Defects and Possible Testing Process Shortcomings

Metric	***1.1.1.1.1 Analysis of Testing/Development Process***
Defect Mode	"Missing" defects is an indication that either not enough customer research took place or that "requirements" did not reach implementation phase. "Unclear" and "Wrong" defects usually means not enough time spent on analysis & design inspections, specification language is vague or customer was not in a position to specify business requirements. "Better way" defects usually reflect lack of design inspections or bad coding practices, while "Changed" defects are an indication of high requirements creep or inappropriate understanding of hardware / system software platform.
Defect Density	Excellent means of quantifying software product quality as well as to assess the effectiveness of the testing process. Many industry averages exist, most of which are in the area of 10-20 defects per one thousand lines of code (KLOC) during system testing, while prior to and including system testing values are up to 200 defects per KLOC [Humphrey 1996]. Rubin reports defect rates from a world-wide survey per industry sector with Aerospace at 4.6 defects / KLOC, Financial at 3.1 defects / KLOC and

	System Software 2.0 / KLOC [3]. Rubin also reports 0.9 defects per function point as a world-wide industry average, while Capers Jones reports 1.75 coding defects per function point [5] [3]. Defect densities should drop ten-fold between testing and maintenance phases [4]. If such trends are not realized between prerelease and postrelease of software, then this is an indication that the testing process is not effective. This metric should also be used in parallel with the percentage of effort spent on testing. Limited testing time allocated will result in few defects detected prerelease and many detected post-releases.
Defect Origin	Probably the most useful metric for evaluating information system product quality. By depicting with bar charts all defect densities per SCI, the project manager can pinpoint the weakest links in the software product. Obviously, weakest links must be retested and perhaps, even redesigned. Defects tend to concentrate on certain portions of the entire information system (pareto principle). Once the defective modules are identified, they need to be test drilled to no end.
Defect Age	A good "barometer" of effectiveness of in-process inspections. High averages for defect age are a clear indication that analysis and / or design inspections are not effective.

Software reliability, the most important constituent of software quality, is a function of code coverage. Therefore, in order to put a handle on software reliability problems, the coders must be in a position to assess code coverage. Typical testing without measuring code coverage only exercises around 55% of the code, while with the use of code coverage instrumentation, this can be raised to at least 80% without excessive additional effort [4]. The insertion of numbered checkpoints in the code (i.e. "Function X, Checkpoint N" with fprintf statements) is an alternative to coders, if automated dynamic analysis is not available through the use of software testing tools.

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

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

4 STAMP - PRACTICAL TEST MEASUREMENTS AT SINGULAR

The STAMP experiment (Singular's Testing and Measurement Process) has as objective to reengineer Singular's testing procedures and was funded by the ESSI Program. The STAMP experiment was the basic cause for a number of organizational changes that took place within the company. More specifically, a way to promote the quality as a major concern of the company was to have the technical division manager and Singular board member assume the position of deputy managing director having the management of quality as the primary responsibility. Singular's board of directors recognized that in order to instill a sense of elite at Singular by producing high quality software products, a full time board member must tackle quality at a strategic level. Having the QA manager handle software quality at the tactical level and the deputy managing director handle quality at the strategic and board level is indicative of Singular's commitment to quality.

The main objective of the experiment was to **increase company's competitiveness through the enhancement of its products' quality**. The introduction of an automated testing procedure helped the company to be more productive and qualitative having the ability to succeed cost reduction and spread its market share. The technical contribution of STAMP experiment to Singular's software development process has made an extremely positive impact. Testing related principles and measurements had been understood in detail and software development projects within the company benefit from the dissemination of the acquired knowledge to non-STAMP Singular software developers.

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

- Defects - Detected defects are classified (logic, data handling, computation, etc.) and registered as either missing, unclear, wrong, changed or better way for each baseline project software module tested. The defect densities for each software module are calculated as the ratio of defects per software size (SLOC). Past testing effort were estimated to be 5 days per 1000 lines of code. STAMP experiment has decrease this number by a factor of at least 20%. This will beneficial impact on the cost of the product.

- Test coverage - Percentage of items (requirements, test features, programs, code statements and branches) covered during testing. During the STAMP experiment an increase of 50% on the test coverage took place. That's because of the introduction, in the testing process, of a testing tool, which offered the chance of a more exhaustive testing.

- Product Reliability - Failure rate and Mean Time between Failures (MTBF) during the test period and after release. The failure rate presented a decrease of 30% while the Mean Time between Failures was increased to 35%.

- Test Analysis - Analysis and study of tests to measure testing effectiveness. The overall effectiveness of testing was appreciated to be increase to 40%. That happened because of the general reengineering of the testing process which set up procedures according which the testing process is taking place.

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

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

- Testing life-cycle phases supported

- ◊ Test Planning
- ◊ Test Design and Creation
  - ⇒ Recording
  - ⇒ Tool Customization
  - ⇒ Script Language
- ◊ Test Execution
  - ⇒ Playback
  - ⇒ Verification, including Static Analysis
  - ⇒ Debugger
  - ⇒ Code Coverage (Dynamic Analysis)

- Report and Analysis
  - ◊ Report Capabilities
  - ◊ Report Customization
  - ◊ Charting Capabilities
- Problem Tracking
  - ◊ Defect Logging
  - ◊ Workflow Tracking
  - ◊ Documentation
- Testing tool Repository
  - ◊ RDBMS which stores the repository
  - ◊ Capability to copy test procedures from one repository of the testing tool to another repository of the same testing tool
  - ◊ Simultaneously multi-user access
- Configuration
  - ◊ Development Environment Supported (e.g. Oracle Developer/2000, Delphi, Visual Basic, Power Builder)
  - ◊ Operating System (i.e.Windows 95, NT, UNIX)
  - ◊ Application-under-test type (i.e. The applications-under-test could be Windows client/server applications or character-based UNIX applications, or Web-base application)
- Ease of Installation and Learning
  - ◊ Installation Process
  - ◊ Tutorial
  - ◊ On-line Help

The following testing tools were evaluated, prior to selecting **Rational's SQA**:

- ◊ Rational's SQA Suite TeamTest Edition,
- ◊ Compuware's QA Run - QADirector - QATrack,
- ◊ Vermont Creative Software's Vermont High Test Plus
- ◊ Segue Software's QA Partner

During the experiment it had been noticed that the set up time of all testing tools including the selected one, was too high in relation with the time of executing the tests. Therefore the usage of the testing tool was not applicable on small-scale projects. Thus the testing tool is used in large-scale projects where its usage, were justified by the complexity of the project.

STAMP has elaborate in detail on the testing tool evaluation process and on the results generated from evaluation of the above listed four testing tools. The testing tool is currently used by 64 users. Since its very first installation 2.389 bugs and 335 errors have been detected.

During the STAMP experiment a categorization of defect types took place. The most frequent type of defects as well as their differentiation before and after the STAMP experiment are listed below.

Table 2: Defects occurrence before and after the STAMP Experiment

Before STAMP Experiment	
Defect Type	***Number***
Back end bug	204
Missing Functionality	114
No Error Found	304
PDC Bug	550
Print Bug	198
UI Bug	286
Other	87

After STAMP Experiment	
Defect Type	***Number***
Back end bug	285
Missing Functionality	133
No Error Found	396
PDC Bug	581
Print Bug	234
UI Bug	304
Other	110

Thanks to the establishment and adoption of the suitable testing methodology the number of the detected defects has increased in a way that permits the release of more qualitative software projects. Moreover the overall reengineering of the testing process turned into a fact the possibility of decrease the mean time for defects correction. While in the past the average time for defect correction was 9.3 working days, now the corresponding time is 7.2 working days.

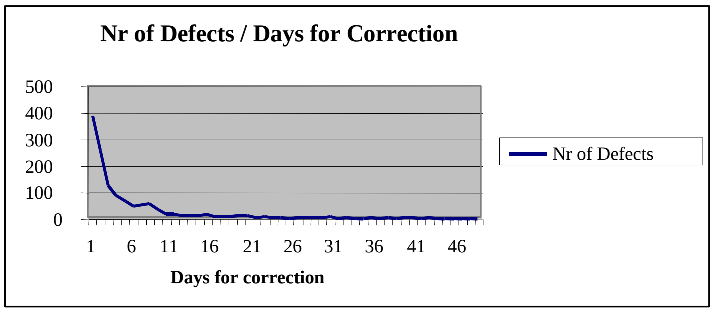


Figure 3: Number of Defects / Days for Correction

The above picture shows the number of defects per days needed to be corrected. While, as it is already mentioned, a greater number of defects have been detected and the mean time for defects correction has significantly decreased, however the total testing time has increased in a small percentage. That's because of the usage of the testing tool that needs a particular effort for its setup for a specific project. The added value gained by the established methodology is focused on the quality of the final product as well as to the economies of scales which are ensured through a common admitted testing methodology within the company.

5. CONCLUSIONS

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

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

6. References

1. Lowell, A: Improving software quality : An insider's guide to TQM, Wiley. 1992.
2. Grady, R.: Practical software metrics for project management and process improvement, Prentice-Hall, Inc. 1992.
3. Rubin, H: Worldwide benchmark project report, Rubin Systems Inc. 1995.
4. Grady, R.: Practical software metrics for project management and process improvement, Prentice-Hall, Inc. 1992.
5. Capers, J:.Applied software measurement, McGraw-Hill. 1996.
6. Capers, J: Revitalizing Software Project Management. American Programmer. 1994.
7. Grady, R.: Practical software metrics for project management and process improvement, Prentice-Hall, Inc. 1992.
8. Hetzel, B: Making software measurement work : building an effective program, QED Information Sciences, Inc. 1993.

Description of the Company/Business/Product

Singular S.A is a well-known company established in Greece on 1984. The company has shown a tremendous growth having presence into 25 countries through its subsidiaries and affiliated companies. Singular has a customer base exceeding 30.000 customers all over the world. The company is mainly focused on two main categories of activities. The enhancement and trading of its products and the provision of customized IT solutions.

The company employees more than 600 persons with a prospect to reach 2.000 employees in the next three years. This is because of the rapid expansion of the company not only in the Greek IT market but also within the International market as well.

One of the two main sectors of the company is the Products sector, which was the initial cause of the company's strengthening and expansion. The company has develop a number of products covering most the Financial and Resource Planning needs of the Greek Enterprises and Organizations. Some of the most famous products of the company is the SEN (Singular ENterprise) ERP system, EUROFASMA which is a product covering the needs of a company's accounts department and the MANAGER a product which covers the commercial department of a trading company and more.

The other main sector of the company is the Projects sector. The objective of that sector is to provide qualitative customized IT solution to the company's clients. The Project's sector has undertaken several large scale IT projects not only for major public and private organizations in Greece but also for major clients abroad.

Authors' CVs

Stelios Pantelopoulos M.Sc.

Stelios Pantelopoulos is employed, since 9/1999, as a project co-ordinator in the Advanced Technologies and Business Development Department of SINGULAR S.A. and is currently involved in the management of running European projects.

As a Senior Software Engineer, for Intrasoft S.A (09/1997-09/1999) he was involved in the analysis and design of database management systems (e.g. CORDIS: the official EC web site for the dissemination of R&D information, PERICLES: software managing the allocation of the rooms and the assignments of the interpreters in the meetings of the European Parliament, etc). During that time he was partially involved in the technical management of European R&D projects (e.g. PROTONET: implementation of an electronic commerce platform for SMEs, TAPPE: development of an integrated system for the public procurement process, etc).

As a Software Engineer for Athens Technology Center S.A. (09/1994-09/1997) he was involved in the software implementation and database development of client server applications (e.g. multimedia database applications, electronic catalogues, software tools for archaeologists, etc).

Mr. Pantelopoulos has over five years of experience in participating in the implementation of IT projects and over three years of experience in managing multi-people teams, while having a good understanding of business issues. He is also active member of the Technical Chamber of Greece.

Yannis Panopoulos M.Sc.

Yannis Panopoulos is employed, since 1/2000, as a project co-ordinator in the Advanced Technologies and Business Development Department of SINGULAR S.A. and is currently involved in the management of running European projects.
As an IT consultant for the Municipality of Aegaleo he was responsible for the reengineering of the IT processes of the municipality as well as for the municipality’s IT procurements.

As a member of the research unit of the Athens University of Electronic and Business (Hellenic Electronic Trading UNit) he was involved on several e-commerce project as a project manager.

As Software Engineer, for Intrasoft S.A he was involved in the analysis and design of database management systems (e.g. ERTICO a project for the European Community and T.E.E for the Technical Chamber of Greece).

Mr. Panopoulos has over four years of experience in participating in the implementation of IT projects and over two years of experience in managing IT projects, having a solid background on business issues.

Session 7 - SPI and Testing

Session Chair:

Carsten Jorgensen,

Delta

Automated Testing for User Interfaces of Engineering Analysis Software, ESSI Project DATES

Graham Spence, Sarah Phillipson and Giles Moran
AEA Technology, 8.19 Harwell, Didcot, OX11 ORA, UK.

Introduction

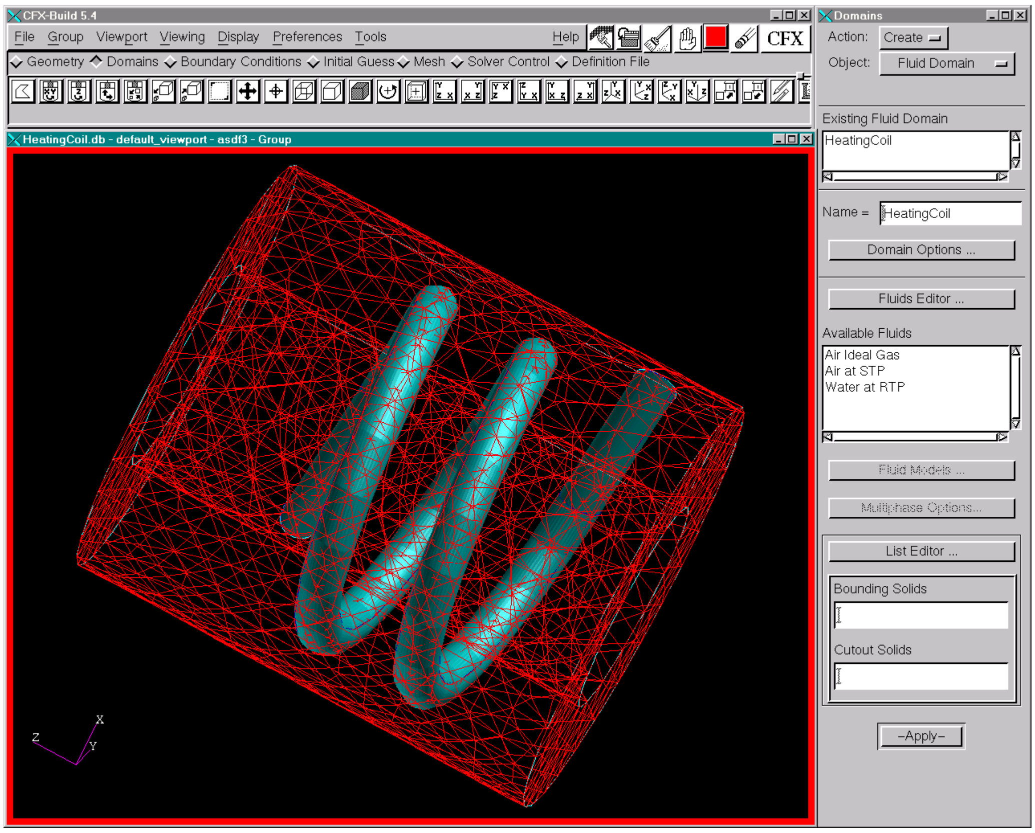
In the face of competition, business success for developers of Computer Aided Engineering (CAE) software relies crucially upon improving the functionality and quality of the product through new releases as quickly as possible. Software testing procedures have been identified as an area in which improvement can significantly reduce the product release cycle.

Existing tools and techniques for automated Graphical User Interface (GUI) testing have not been properly evaluated and demonstrated in the context of CAE software, and this inhibits the adoption of automated GUI testing for CAE. Most CAE software packages make significant use of 3D graphics on both Windows NT and UNIX platforms, which places particular demands upon automated GUI testing tools and procedures.

As part of a Process Improvement Experiment funded by the European Software and Systems Initiative, AEA Technology has developed a test suite that demonstrates the automated testing of a CAE GUI, in a project entitled „Demonstration of Automated Testing for Engineering Software“ or „DATES“. This paper describes the process of creating such a suite and the evaluation of some of the results.

Background to the experiment

The DATES Process Improvement Experiment has been performed during the development of the latest version of CFX-5. CFX-5 is a CAE software tool for fluid flow simulation which relies on interactions between the user and the graphical user interface to allow the easy set-up of a simulation. Typical interactions of the user


CFX-Build 5.4
File Group Viewport Viewing Display Preferences Tools Help
CFX
Geometry Domains Boundary Conditions Initial Guess Mesh Solver Control Definition File
HeatingCoil.db - default_viewport - asdf3 - Group
Domains
Domain Options ...
Fluids Editor ...
Fluid Models ...
Multiphase Options...
List Editor ...
Apply
X
Y
Z

Action: Create
Object: Fluid Domain
Existing Fluid Domain: HeatingCoil
Name = HeatingCoil
Available Fluids: Air Ideal Gas, Air at STP, Water at RTP
Bounding Solids:
Cutout Solids:

Fig. GTS.1 : A screenshot of the CFX-5 software. The dark area is the „viewport“ which contains a representation of the model used for the simulation.

with the graphical user interface include the selection of various objects in a drawing area („viewport“), and this presents a major challenge when it comes to applying a GUI testing tool to this interface, since without a correct selection being made, the set-up of the simulation cannot proceed. A screenshot of CFX-5 is shown in figure GTS.1. Additional complications are due to multi-platform support (CFX-5 is supported on five UNIX platforms and Windows NT) and to the number of different components with graphical user interfaces, which are written in different programming languages and based on software from different suppliers.

The primary aim of the experiment has been to quantify the benefits to be gained from the automation of the testing, and to use automatic GUI testing to reduce the time required for each new release by improving the software testing procedures. This should be achieved by doing much more regression testing during code development, and speeding up the final acceptance testing.

Process Improvement Experiment description

The overall plan for the experiment has been as follows.

- Evaluate the existing automated GUI testing tools in order to choose which is the most appropriate for testing CFX-5.
- Acquire the tool and gain experience of its use.
- Define new working procedures to include significantly more regression testing during development taking advantage of the automated GUI testing tool.

The aims of the experiment have been as follows.

- To demonstrate a 15% reduction in total project elapsed time for a CFX release, as a result of the changes in procedure made possible by automated GUI testing.
- To obtain a measurable improvement in product quality and, in the long term, to reduce development costs.

To enable the appropriate results to be measured, the DATES project has considered two software projects within CFX-5 development. The first, CFX-5.3, was completed without any use of automated GUI testing software. The second, CFX-5.4, has been completed with use of the GUI testing software throughout most of the development and testing stages.

Development Process for CFX-5

To put the development of an automated GUI test suite into context, it is necessary to consider the development process used for CFX-5.3. The main stages of development included requirements specification, design, coding, user documentation and pre-formal testing, bug-fixing, final acceptance testing and then release. There was some overlap between these stages. During the coding stage and subsequent stages, there were automated nightly rebuilds of the developing software, on all target computer platforms, starting as early in the development cycle as possible. These rebuilds included basic automatic testing to ensure that all components work together, and this meant that the development team was aware of major porting problems as soon as they occurred. The acceptance testing stage consisted of substantial amount of manual testing on each platform, according to a detailed test plan, in addition to the re-running of the automatic tests.

However, the automatic testing for CFX-5.3 did not include any testing of the graphical user interfaces. This meant that there was significant potential to find significant problems in the user interface very late on in the development cycle, often in the acceptance testing stage, when defects are very expensive to fix. This had to be compensated for by ensuring that the coverage of the GUI in the manual testing was very thorough.

The hope for CFX-5.4 was that automatic GUI tests could be included with the nightly rebuild automatic testing, on each platform, so that bugs could be found in the user interface as early as possible, when they are cheap to fix.

Evaluation of the existing GUI testing tools

The evaluation of the existing GUI testing tools was performed by a third-party consultant, mainly by using the World Wide Web to gather information. Our requirements for a testing tool included the following, and this brought the number of potentially suitable testing tools down to just three.

- The chosen tool must be supported on both UNIX and Windows NT platforms, or two very similar tools must be available.
- Various requirements for tests to be robust when the GUI changes slightly or when the display is changed indicated that the chosen tool must be able to work in a „widget-based“ mode (where mouse-clicks operate on specified widgets, rather than at a specific location on the screen).

More details can be found in reference [1]. A more general overview of the use of automated GUI testing tools can be found in any recent textbook on automated testing (in particular, see reference [2]) or see, for example, reference [3].

The three short-listed tools were then evaluated in more depth against the detailed requirements that the chosen testing tool had to satisfy and the final choice was made based on how the tools performed on the CFX-5 software during the evaluation. The chosen software was the XRunner/WinRunner software from Mercury Interactive.

Creating the test suite

As would be expected, the creation of the test suite took a considerable investment of time, even apart from the time taken in learning how to use the chosen GUI testing tool. As a starting point, as much of the acceptance testing procedure as possible was automated, and then other tests were added to increase the coverage of the testing. This involved developing solutions to a number of potential difficulties, as described below.

- On UNIX systems, in order to select certain objects reliably from the drawing area („viewport“), it was found necessary to run the tests using a virtual display. This means that the tests run in a consistent environment independent of which machine runs them and which display is used. This also allows multiple copies of the tests to be run simultaneously without access to multiple real displays. Unfortunately, this did not prove to be successful for one UNIX platform, and as a result, the tests on this platform had to use a different set-up. However, tests on the other platforms run reliably using the virtual display.
- One component of the software, CFX-Build, currently uses the Exceed software (from Hummingbird) to provide an X Windows display on Windows NT. As a consequence of this, it did not prove to be possible to test CFX-Build on Windows NT. This was a significant limitation of automatic GUI testing on Windows NT.
- Three minor components of the software are written in Tcl/Tk, which is not fully supported for the chosen automated GUI testing tool. As a result, although some tests could be developed for these components on Windows NT (where limited support was available), it was not possible to develop tests for these components on UNIX systems using the „widget-based mode“. It was not felt that it was worth spending time attempting to test these components on UNIX in „position-based mode“ due to the high maintenance cost of such tests.

Despite these limitations, it was possible to develop a test suite for some parts of the software that would run on both UNIX and Windows NT without substantial re-recording of tests, and that would run reliably. Most of the rest of the software was covered on either UNIX or Windows NT, but not both. In order to check whether or not the GUI was behaving correctly, three types of comparisons could be performed during the test execution: either checking the appearance of the model in the viewport of the software (which used a bitmap comparison facility provided by the testing tool), checking whether or not the correct files had been written at the end of a test, or checking whether the software acknowledged the data it had received by (for example) displaying a particular form or message. In most cases, a combination of the three were used. The maintenance of this test suite and the bugs it found are discussed below.

After its development, the test suite was integrated into the existing regression test structure, so that it was accessible to all developers without any special knowledge. This helped a great deal in improving the visibility of the GUI tests and ensuring that they were run often. This was, for the most part, very successful. Part of the work undertaken for this step was to write a small piece of code to parse the reports produced by the GUI testing tool to present information in a form which is more readily readable by a developer not trained to use the tool itself. As a result, the developers can now run the automated GUI tests using a structure which is familiar, and get the results in a form that they can read and understand. This includes side-by-side pictures of expected and obtained results in places where a bitmap comparison has failed, which makes it very easy to see where the difference lies.

Unfortunately, it did not prove to be possible to integrate the Windows NT test suite into the existing structure (which makes heavy use of UNIX scripts), and the Windows NT tests have to be run separately in consequence. Various problems were also found in integrating the UNIX tests on the one platform which could not use the virtual display, and these also have to be run separately.

Maintenance of the test suite

The maintenance of an automated GUI test suite can prove to be a time-consuming task. However, since all of the tests in our test suite use the widget-based mode, this means that the tests are robust against the addition of new widgets to the GUI or changes in widget placement. The main maintenance tasks have proven to be:

- updating the GUI information used by the automated GUI testing tool when widgets or forms change their names or types as the software is developed.
- updating the images of the viewport which are used for comparisons. These can change due to operating system changes, upgrades to versions of the graphics software and other similar changes.
- restarting the virtual displays used for the tests.

The test suite proved itself able to find various types of bug, some of which would not have been found even by performing the same test manually, unless the tester had extremely sharp eyes. Perhaps the best example of such a bug is one that stopped the image of the coordinate axes (located at the bottom of the drawing area – see Fig. GTS.1) rotating with the geometry. Other bugs found included a hard-coded path reference in one of the components which can only be regression-tested using a GUI testing tool, a variety of bugs which prevented the data entered through the GUI from getting to the correct set-up file (which usually showed up by a failure of the software to write the file correctly), or incorrect manipulation of the model in the viewport (which showed up when doing bitmap comparisons of the viewport image).

Results of the experiment

Some quantitative results can be obtained from the experiment by comparing the two projects, CFX-5.3 and CFX-5.4. We need to proceed with some caution in the generation of these statistics due to considerable differences in the projects, and some thought needs to be given to how best to provide a „normalisation“ of the statistics, given the differences between the two projects.

CFX-5.3 was under development for approximately 12 months. It was supported on five UNIX platforms only – a port to Windows NT was performed after the CFX-5.3 release on UNIX, and the statistics for this porting project have not been included in the analysis presented here. The project was released on schedule. The project included two new major pieces of functionality, but the underlying code was relatively unchanged.

CFX-5.4 was under development for approximately 18 months. It is supported on five UNIX platforms and Windows NT. The project included four new major pieces of functionality, which included an entirely new component, and substantial re-work of two of the largest components. The source code is approximately 1.5 times as large as it was in CFX-5.3 (measured in number of lines of code).

It is clear that before meaningful comparisons can be made between the two projects to assess the use of the automatic GUI testing, we need to take into account the other differences between them. This has been done by scaling or „normalising“ the figures as described below.

When considering the final acceptance testing, a sensible normalisation would be to take results *per platform* and to scale by *project size*. During final acceptance testing, almost all testing is performed on every platform, so clearly in terms of total man-hours one would expect the amount of testing required to achieve the same coverage would scale approximately linearly with the number of platforms. (Note, however, that the addition of an extra platform would not be expected to change the total *elapsed* testing time significantly.) It is also clear that to achieve the same degree of confidence in a larger product requires more testing (since there is more to cover), so normalising by size of the product is the obvious choice here. Here, we take number of lines of source code to give a suitable indicator of the product size.

Reduction in release time

The main stages in the development process are requirements specification, design, coding, acceptance testing and release, as described earlier. Clearly, the only stages at which the introduction of automated GUI tests would be expected to reduce the time taken is in the coding and acceptance testing stages.

Considering the coding stage of the development, one might expect that the introduction of automated GUI tests could help by finding bugs earlier, when they are cheaper and less time-consuming to fix. For example, if a bug is detected the same day as it is introduced, then it is very easy for the developer to fix the bug immediately, but several weeks later it would be very much more difficult to track down precisely the change which caused the bug. However, time savings from this are very hard to quantify. The following problems also prevent any convincing demonstration of reduction in development time.

- The two major components with a GUI were already regression-tested by the use of „session files" to give the component its instructions. Most of the bugs in these components are not solely GUI bugs, and so would already have been picked up by our existing automated tests.
- Of the three more minor components with a GUI, very few changes were made. So whilst running the GUI tests gave us confidence that these components were still in a good state, the tests did not uncover a large number of bugs.
- The automated GUI testing suite we developed was a regression testing suite, that is, it only tested parts of the GUI that had already been tested manually (in fact, the very creation of an automatic GUI test is in itself a manual test of the GUI). Therefore, since the majority of new bugs occurred in the new parts of the product, the automated GUI tests did not account for a large number of the bugs found. Most bugs are not platform-specific, so the advantages of running automatic GUI tests on all platforms are not especially apparent.
- There are so many different combinations of panels that can be achieved with the CFX-5 software that the coverage of the automatic GUI tests could not be high. Since there are many panels which every user must go through every time they use the software, the manual testers would catch many of the GUI bugs in these panels anyway.

Considering the acceptance testing stage of the development cycle, there are more promising ways to look for a reduction in the time taken. Firstly, the acceptance testing stage involves a set of very clearly defined tests designed to give us confidence that the product is fit to ship, and secondly, these tests are repeated on every platform. These two criteria (well-scripted tests which are run many times) are key in determining whether automated testing is likely to prove beneficial – see reference [4].

Clearly, the final testing stage is crucial to ensuring that CFX-5 is of a high enough quality to be shipped to customers, and we cannot afford rely unduly on new and relatively-unproven automated GUI testing technology. Nevertheless, after experience gained during the coding stage of the development cycle, we felt confident enough to be able to reduce the amount of manual testing performed during the acceptance testing stage by running the equivalent automated GUI tests. Figure GTS.2 shows how the amount of manual testing performed in CFX-5.3 and CFX-5.4 compares.

The bulk of the time spent doing manual testing in CFX-5.3 was to cover the

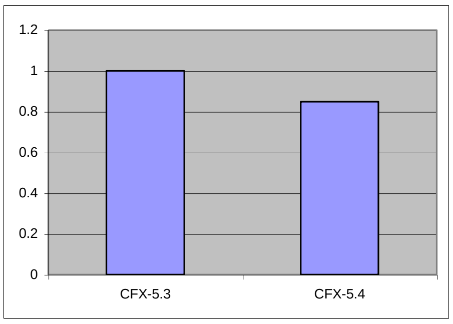


Fig. GTS.2 : Manual testing required for final acceptance testing, in days per platform, normalised by the total project size

installation and the GUI; the major part of the testing of the GUI involved running through our tutorial material since that covered all the major features in the GUI, and also provided a check that the tutorial documentation was of good quality. It was not possible to automate the installation procedure efficiently using the automated GUI test suite because the installation is very platform-specific and cannot be run often enough to justify the effort involved in producing the automatic tests. The bulk of the GUI testing was automated for CFX-5.4; however, significant effort still needed to be spent doing manual testing on the tutorial material in CFX-5.4 since the automated GUI tests cannot test that the tutorial documentation still makes sense and is easy-to-follow. Hence the reduction in manual testing shown between CFX-5.4 and CFX-5.3 is not so great as would perhaps be expected. For future releases, there is potential to reduce the manual testing still further, as we gain experience and confidence in the use of the automatic GUI testing tool.

Increase in product quality

Product quality can be measured directly in terms of numbers of bugs. However, the released product quality can really only be measured by looking at the bugs found by customers. At the time of writing (July 2000), CFX-5.4 has not been on release long enough to gather useful bug statistics from customers.

Other results

The existence of the automated GUI test suite has improved our confidence that the GUI of the CFX-5 software is in a good state after each nightly rebuild. This reduces the time wasted when a tester manually starts up the software, only to find that the GUI has broken. If the automated GUI tests show that the GUI has broken, then this can be fixed as a matter of priority, and testers can get working again in a shorter time span.

Conclusions

- An automatic GUI testing tool for CAE software which makes use of 3D graphics and the selection of graphical objects on multiple platforms (including both UNIX and Windows NT) has been employed successfully and integrated in the development program of the CFX-5 software.
- The development and maintenance of such a testing suite is made simpler by making use of a virtual display to play back the tests.

- The testing suite has been integrated with existing regression tests (which do not test the GUI) for ease of use.
- The manual testing time has been reduced significantly by the use of such a suite. It has not yet been possible to gather enough statistics on the improvement obtained to quantify the results reliably.
- The use of the automated GUI test suite has helped to improve the predictability of the CFX-5.4 testing process by increasing confidence in the quality of the GUI during both the development stage and final testing.

References

[1] Stephen Morris, GUI Testing Tools Evaluation, http://www.software.aeat.com/cfx/European_Projects/DATES/bestprac.html

[2] Mark Fewster and Dorothy Graham, „Software Test Automation", Addison-Wesley, 1999.

[3] Brian Marick, „Classic Testing Mistakes", http://www.testing.com/writings/classic/mistakes.html

[4] C. Kaner, „Improving the Maintainability of Automated Test Suites", http://www.kaner.com/writings/lawst1.htm

Company Information

CFX is AEA Technology's powerful fluid engineering software that simulates industrial flow and heat transfer processes, providing new insights and better understanding of the most complex flows. A broad range of industries and research institutions uses CFX for modelling and simulation in the design and development of sophisticated products and processes.

AEA Technology Engineering Software is a leading supplier of engineering solutions for the process and mechanical manufacturing industries. The business operates from a network of 20 offices worldwide, with subsidiaries in Canada, USA, Germany, Spain, France and Japan. Its range of software products includes computational fluid dynamics (CFX), process simulation (HYSYS), separation process design (SPS), batch process design (BaSYS & BDK), heat transfer (HTFS), plant lifecycle data management (AXSYS), and fatigue and lifetime prediction solutions (nCode). The business aims to provide an unequalled software environment for the management of knowledge and design throughout the engineering and operational lifecycle within its expanding customer base.

AEA Technology Engineering Software is a business of AEA Technology plc, one of the world's leading innovation businesses supplying innovative technical, safety and environmental solutions to a broad range of industries worldwide. AEA Technology plc employs nearly 4,500 people in 26 locations around the world, bringing science to the marketplace through consultancy, technical services, hardware and software systems, research and development and technology transfer.

Author Information

Graham Spence

Project Role: Test Engineer responsible for the DATES project work

Selection of Project Experience:

- 1998-2000. DATES project work, including setting up the testing suite, writing reports, some project management.
- 1998-2000. As a member of the CFX-5 development team, participating in manual testing and providing feedback on the usability of the software.

Qualifications:

- BSc, Computer Science with Artificial Intelligence, University of Leeds.
- PhD, Department of Fuel and Energy, University of Leeds.

Sarah Phillipson

Project Role: CFX-5 Testing Manager

Selection of Project Experience:

- 1999-2000. Planning, organising and managing the testing of CFX-5, including testing through development and final acceptance testing.
- 1999-2000. Setting up a system for collecting validation cases for CFX-5, including automatic testing of cases to ensure that the software continues to achieve correct results and dissemination of results.
- 1997-1998. Writing the user manual for a new part of CFX-5, the visualisation program CFX-Visualise.

Qualifications:

- BA, Physics, University of Oxford.
- PhD, Theoretical Physics, University of Manchester.
- Foundation Certificate in Software Testing, issued by the ISEB.

Giles Moran

Project Role: CFX-5 Pre-Processor Developer

Selection of Project Experience:

- 1998-2000. Development of CFX software. Involved in the design and implementation of the CFX-5 Pre-Processor.
- 1998-2000. As a member of the CFX-5 development team, participating in manual testing and providing feedback on the usability of the software.

Qualifications:

- BSc, Physics, University of Wales.
- PhD, Department of Physics, University of Wales.

Promote: Metrics & Models to improve the test process

Traversa Mauro
Elsag spa, Genoa - Italy

Abstract

This article reports the Process Improvement Experiment (PIE) performed in Elsag spa, a large italian company whose main business is in postal and services automation, with the main goals to improve the development and test processes.
Lack of standardisation and rules in the programming step (C/C++ languages), bad-structured and too complex routines, and high costs and times for the test activities performed without criteria and measures of coverage, have been the main problems to be fixed in the PIE.
The experiment (June 1998-March 2000) has been performed in collaboration with CEFRIEL, an italian consortium with a large experience in the field of testing metrics whose role was to select metrics and to calculate and tune the statistical models used in the PIE
Working, as case study, on a critical software component (the recognition address task on a postal object), we:

- selected a set of potential code violations to be filtered in our software with the target to introduce a more reliable style of programming;
- selected classic metrics of complexity to analyse and reduce the structure of the routines;
- defined and tuned statistic models (the static and the dynamic models) calculating, for any routine, the probability to have faults before and after testing and estimating the minimum number of useful test cases to execute.

The results of the experiment have been quite positive and the technological, the economical and cultural impacts have been measured. After the conclusion of the best practise, Elsag is extending the new process to other significant software components.

The Background Scenario

Elsag spa is a large engineering company mainly operating in the industrial sectors of postal automation, image processing, document management, and automation of services. The software components mainly embedded into more complex systems contribute to the total business of Elsag for about 180 MECU/year and represent about 50% of the company business. This percentage constantly increased in the last 10 years and it is expected to rapidly increase in the next years.

Company Organization

Elsag organization is based on a Central Development and Research Department (SRS), where activities of development and test are concentrated, interfacing with several Business Units whose responsibility is to manage the projects directly with the external customers.

The SRS Department, where the PIE has been performed, has the following organization:

- A team responsible for development and unit test of single modules
- A team responsible for performing system and integration test
- A team responsible for software configuration management activities

The development team (about 40 members) works mainly on Window NT platform and uses Microsoft Development Studio (Visual C and Visual Basic) to develop the software components.

The test team (about 12 members) uses Test Director by Mercury as database of test cases and Winrunner by Mercury for the validation of the Graphical User Interfaces.

The test team (for the testing phase) and the business units (after the delivering of the products) use PVCS Tracker to report the detected anomalies while PVCS Version Manager is used by the Software Configuration Management Team (3 members) for change control and release management activities.

Products

Elsag operates in different technological markets with a large catalogue of several products. One of the most strategic areas where the company is actually investing is the market of the postal automation. Elsag proposes to its customers (that are typically national postal organizations) complete hardware and software products for automatic sorting of postal object where for postal object we consider letters, magazines, boxes,…. Actually we have many systems working in Chile, Indonesia, South Korea, Japan, Kuwait and Taiwan. In Europe, Elsag is the main contractor of the Italian Postal Organization and is now selling and installing 15 systems to France.

The key factors of a postal machine are the level of reliability of hardware and software components (supposed to work 24 hours a day), the amount of postal objects sorted for hour and the percentage of automatic recognition of the addresses located on the postal object. This last factor is very relevant and lets the customer tune the number of human resources necessary to complete manually the address recognition task. The case study considered in the PIE regards this strategic software component.

The Starting Scenario

This section describes the problems and needs we identified in our organization, the goals we planned to achieve in the PIE and the expected benefits we estimated to reach after the conclusion of the experiment. The last subsection describes our selection about the case study.

Identified Problems

We were not able to diagnose our organization against well-known models (CMM, SPICE,..) and we had no a great amount of quantitative measures really demonstrating the weak points of our process. Essentially, to identify our problems, we organized many interviews with the middle management and with the development and test teams. The result of our analysis was the following one:

- Every developer had his own style of programming; standardization was not well defined and accepted. Consequently, the software delivered to the test team was not homogeneous and often presented an unacceptably high variety of bugs and violations.
- Too often, fixing cosmetic bugs and minor improvements generated instability and side effects on the behavior of the software component.
- Testers did not have explicit criteria to determine the termination of testing activities. No coverage was currently in use and the black box approach limited the number of significant information useful to increment this level of coverage.
- Test cases are expensive to be executed and managed. We had evidence that the whole set of test case currently used to validate our software was not necessary: a significant number of test cases may be eliminated without significantly affecting the quality of the product. The possibility to determine the minimum set of test case necessary to guarantee the maximum level of coverage could reduce the costs.

Technical Objectives and Benefits

Considering the problems listed in the previous subsection, we identified the following objectives:

- The introduction of rules for developers with the goal to define a standard and a, as more as possible, homogeneous style of programming. These rules are essentially based on the selection of C/C++ language constructs that are potential cause of bugs (we call them „violations“ according to the nomenclature used by the tool of source code analysis used in the experiment).
- The introduction of metrics to measure the level of complexity of our software and to identify, with the support of the selected tool for the experiment, the best criteria to reduce this level of complexity under the defined thresholds.
- The introduction of statistical models whose goal is to identify the most critical components to be tested and the indices of code potential faultiness of software components before the delivering to customers.
- The introduction of statistical models estimating the minimum number of significant test cases necessary to assure an acceptable level of faultiness of the delivered software.
- Reaching the previous objectives, automatically we plan to reach another significant high-level objective: the introduction of the culture of software metrics in our process to measure it and to understand which are the strong and the weak points of the process itself.

The expected benefits were:

- Software easier to be managed in terms of maintenance and testability.
- Significant reduction of time and costs of system and integration testing.
- Reduction of the number of the test cases obtained by eliminating redundant tests.

The case study

The selected case study is one of the core products of Elsag spa: automate character recognition system, named OCREngine.

OCREngine is a powerful, scalable and flexible family of products for automatic postal object reading. It is based on a common base technology covering a wide area of possible applications, wiping from form processing (for document processing applications) to postal object automatic encoding based on address interpretation (for postal automation applications).

A simple estimation of the size and complexity of this product can be derived from the following table:

Number of source modules	≈ 600
Number of code lines	≈ 115.000
Size of source code	≈ 4.6 Mbytes

The OCREngine system software has a strategic rule of the product in Elsag spa, and we can affirm that many customers really evaluate the opportunity to buy our postal system according to the quality of our address recognition task for a postal object. The percentage of recognized addresses is typically a basic requirement in the customer specifications and it represents always a basic acceptance test for our customers.

Elsag selection for the PIE is focused to a software module whose task is the identification of any single hand/type written character of the address on a postal object. This software is composed of 30 routines (1180 Lines of Code) written in C/C++ language

Metrics & Models

From the technical point of view we can say that we worked on four different areas with the objective to achieve all the planned goals. The first two areas were investigated and managed with the collaboration of the development team, while the third and the fourth ones were investigated with the collaboration of the test team.

Violations

With the target to reduce the number of C/C++ languages constructs that are potentially cause of faults we selected and classified, three different classes (A, B and C) of violations. These violations have been selected by our developers in the set (more than 100) of suggested ones by the commercial tool that we used in the experiment. The tool is able to detect and to report any critical language construct when the source code analysing functionality is running. Our development team analyzed any suggested violation and selected the most significant ones according to the frequency they were occurring in our software and according to their level of gravity.

- In Class A we inserted mandatory violations and software has to be modified by developers to avoid them. This class includes „Procedure name reused“, „Function does not return a value on all path“,..
- In Class B are included constructs written in a not-accurate way that can be modified to improve the readability and the maintenance of the routines. This is the case of „No brackets to loop body“, „No brackets to then/else“, „Infinite loop used“,…
- In Class C there are constructs that we do not consider real violations but we plan to filter them. In Class C we have „Use of break statement in loop“, „Use of continue statement“, Use of shift operator on signed type“ and similar.

Complexity

With the target to reduce the complexity of the structure of code we focused on two well-know metrics:

- Essential complexity
- Essential knots

In this case the target has been to concentrate the attention of the developers on critical (from the complexity point of view) components and give to the test team useful indications about the complexity of the software to be tested.

The Static Model

Using techniques of Logistic regression, it has been built a statistical model which is essentially a function of static metrics captured with the selected tools and it represents, for any routine, the probability to have faults. This information, captured before the testing phase, is used by the test team in costs evaluation and let the test activity to be focused on modules with a high probability to have faults.

This probability is defined as:

$p(faults)=\pi(FP, CONTROL, V(G), LOC, Lines, Comments)$ where

$log(\pi/1-\pi) = - 4.76740505 + 0.4789582\ FP -0.68208893\ CONTROL + 0.6643859\ V(G) - 0.50226561\ LOC + 0.4514182\ Lines - 0.42495702\ Comments$

where *FP, CONTROL, V(G), LOC, Lines, Comments* are classic static metrics [1][2].
See Appendix 3 for a description of these metrics and criteria to evaluate the goodness of the model.

The Dynamic Model

Always working with techniques of logistic regression, it was built a second model defining the probability that a fault remains uncaught despite the achievement of a given coverage. This model is to be applied to any routine after the conclusion of the testing phase and give useful information about the potential residual faultiness of the delivered software and indications about the minimum significant number of test cases. Using coverage measures captured with Testbed, it's possible, for any routine, to define, x as the ratio between the number of test cases that detect the failure (revealing test cases) and the number of test case exercising the routine (passing through test cases). The range of values of the reveal/pass-through rate has been divided in four distinct sub-ranges identifying five distinct classes (from Class 0 to Class 4) representing different level of faultiness:

- Class 0: non-faulty modules,
- Class 1: $0<x<0.054$
- Class 2: $0.054<x<0.18$
- Class 3: $0.18<x<0.66$
- Class 4: $x>0.66$

We estimate that the number of faults still present in a module after testing decreases moving from Class 1 to Class 4. We define again the probability *p(i)* of a module to belong to Class *i* as:

- $p(0) = 1/(1 + e^{-5.3820388 - param})$
- $p(1) = 1/(1 + e^{-6.5149607 - param}) - 1/(1 - e^{-5.3820388 - param})$
- $p(2) = 1/(1 + e^{-7.4821628 - param}) - 1/(1 + e^{-6.5149607 - param})$
- $p(3) = 1/(1 + e^{-8.971639 - param}) - 1/(1 + e^{-7.4821628 - param})$
- $p(4) = 1 - 1/(1 + e^{-8.971639 - param})$

Where *param* is a combination of well-know static metrics [1] [2]:

param = 1.12013763 eLOC + 0.31751306 Comments – 0.4334234 Lines – 1.204101 FP + 6.06374051 CO – 6.27636 LFC + 8.29360992 eV(G) + 0.32320117 OC – 3.8617338 NEST – 0.9116793 n1 – 6.629129 N +0.44606073 n – 0.0298177 V + 0.00011847 E – 5.4163831 BRANCH – 3.239876 ANION –3.3306005 EXEC + 2.23758035 QCP_MAINT

See Appendix 3 for a description of the metrics mentioned in the above formulas and criteria to evaluate the goodness of the model.

We estimate that a module belongs to the class with highest probability in the above formulas and consequently we evaluate its level of potential residual faultiness.

Further it's possible to estimate *n* as the minimal number of test cases that have to cover each routine to find a fault, if present in the module itself. According to the definition of the reveal/pass-through rate this number is defined as $1/x_{min}$ where x_{min} is the lower bound of the reveal/pass-through rate of the modules belonging to Class *i* (i.e.: if a routine belong to Class 2, x_{min} is 0.054 and consequently *n*=19). The definition of *n* is rather intuitive but to understand better this, the limit situation where any test case passing through a routine detects a fault may be considered. In this case the rate is equal to 1 and 1 is again the minimum number of test cases: really it's logical to think that, if any test case is able to detect the fault, just one test case is necessary to be executed.

We estimate to terminate testing when, for any routine, the number of passing through test cases is greater than *n*.

The Experiment

This section describes the main phases of the experiment performed jointly with development and test teams whose collaboration and work were strategic to determine the success of the experiment itself.

The technical phases of the experiment

The first technical step has been dedicated to the selection of the commercial tool and to the definition of the most suitable metrics and models. Significant internal benchmarks were used to compare different commercial tools and LDRA TESTBED was selected for its high level of performance. Other two tools, Krakatau (www.powersoftware.com) and RSM(www.m2tech.net), were introduced in the project to capture metrics not supported by TESTBED.

After the definition of metrics, models and tools, the project entered in the most two significant technical phases.

The first one may be defined as the first experience phase (the offline phase). We used the tools to captures the selected measures on a delivered version of the case study with the target to „take a photograph" of the initial state of art in terms of violations, complexity and level of faultiness of the delivered software.

At the end of this task we defined a first version of the new process applicable to the current industrial process and developers had useful information to improve their software in term of violation and complexity.

These improvements have been introduced in a new release of the case study previously planned with our customers for bug fixing and minor technical enhancements. We allocated an extra-budget of time (30% more than the originally planned one) to let our developers introduce the improvements required by the new process. It's important to underline that we applied the new process working in a „realistic industrial scenario" (the online phase) where time for delivering was one of the typical constraints of any product roadmap and the introduction of the improvements deduced in the offline phase was only one of the objectives of the new release. We preferred to do so with the goal to simulate the real scenario of any possible future extension of the new process to other software components where industrial constraints of time and cost must be considered.

After that the improvements were introduced and the new release of the baseline project was delivered to the test team, the last technical phase could officially start. In this last step we captured again the selected metrics and we measured the real level of improvements we reached in terms of violation and complexity. We applied the static model to the delivered software with the goal to give to the test team accurate information about the routines whose probability to have fault was higher to let the test activities to be focused on these. At last, while test activities were running and measures of coverage could be captured, we applied the dynamic model calculating the probability to still have faults after delivering and calculating the minimum number of useful test case to assure a significant level of coverage. Both informations gave a great help to define a criterion to stop test activities on the new version of the case study.

The Results

This table summarizes the state-of-art before the improvements and the results of the experiment. I remember that the case study was based on 30 routines written in C language.

Before the improvements	After the improvements
Any routine had violations included in Class A & B	No violations in Class A. 6 routines with violations in Class B.
36% of the routines with an essential cyclomatic number greater than the threshold (3).	13% of the routines with an essential cyclomatic number greater than the threshold.
37% of the routines with an essential knots number greater than the threshold(0).	16% of the routines with an essential knots number greater than the threshold.
87% of the routines with p(fault) greater than the threshold(0.5) and 23% of these with a value superior to 0.9	45% of the routines with p(fault) greater than the threshold. None with a value superior to 0.9.
13% of the routines have still a high probability to have faults after testing (Class 1) while 23% may be estimated to have a very low risk of failures (Class 4). 20% of test cases do not increase the level of coverage.	7% of the routines have still a high probability to have faults after testing (Class 1) while 61% may be estimated to have a very low risk of failures. Time for test: reduction of 10 %

These models seem to work well in our environment with our software. This can be confirmed for the static model comparing the probability, for any routine, to have faults with the number of anomalies detected in the testing phase. We found a good correspondence between the routines identified by the static model as critical and the reports of the test team: the largest number of bugs was really detected in these routines; on the other side a very short and not-critical number of bugs was detected in the routines identified by the model as faultiness or with a very low probability of faults.

Considering the dynamic model we can declare our confidence in it analysing the first reports coming from external customers about the quality of our software delivered after the introduction of the new process. Again the number of bugs they are reporting decreases and the anomalies tend to be concentrated in the routines classified in Class 1. This is the most significant way to demonstrate the goodness of the model and of the new process we introduced.

Experiment Evaluation

This section wants to give our evaluation of the experiment analysing the technical, the cultural and the economical impacts. Lesson learnt and weak and strong points of the experiment are also described.

The Technical Impact

Violations and complexity metrics do not represent nothing of new or exciting from technical point of view but the reached sensibility about the problems connected to the presence in software of potential bugs (violations) and the reached sensibility toward bad-structured software is something new in our development area. About these points, the most significant technical impact is the introduction of the tools of source code analysis adopted by any member of the development team to capture violations and to have useful information about the complexity of the routines and very useful indication to simplify the level of complexity.

More interesting from technical point of view is the introduction of the statistic models elaborated by CEFRIEL. The experiments performed and the obtained results clearly indicate a high correlation between static metrics (used in both models) and the presence of faults. The detailed statistical analysis performed with different statistical methods indicates that for a homogeneous class of software and a limited set of severe faults it is possible to build a statistical model referring to a reasonable small set of metrics.

The Business Impact

The business impact on software improvements in terms of reduction of violations and complexity is significant but its quantitative evaluation is not very simple in a short period of monitoring. The positive effects of a well written and structured software will be more relevant when, in a short future, this code will be again modified for bug fixing or new enhancements.

Business impacts from the test point of view are more immediate and relevant. The result table reports a reduction of time for test (respect to the originally planned one) of 10% that is a first significant result considering that the case study represents only a small percentage of the software to be tested This has been possible because test activities have been more focused on the functionalities implemented by the routines whose probability of fault was higher. In general this information is very useful also for a better estimation of times and costs for test and the test manager will use it for planning future activities.

Another important aid to reduce time for test is the estimation about the minimum number of test cases useful to be executed. In the experiment it has been calculated that 20% of the test cases do not increase the level of coverage.

At last, we cannot forget the obvious and most relevant impact: the quality of the product delivered to customers. The dynamic model has the goal to calculate, on statistic base, the level of potential residual faultiness of code and consequently gives to the test and product managers a significant help in the critical decisions about the opportunity to deliver the new release. Our products are delivered worldwide (Chile, South Korea, Japan, Taiwan, Indonesia, Kuwait…) and, consequently, costs for bug detecting and fixing in field are very expensive (we estimate about 100 times the cost of bug fixing in house).

The Cultural Impact

The most significant cultural impact is the introduction of the concept of metrics as important aid to measure the quality of our lifecycle process.

Before this experiment we had no rules for a better programming and no sensibility about the significant benefit of a well structured and easy to be maintained software. Different was the scenario in the test area; before the experiment any testers needed criteria to understand if they were using with the best profit the time dedicated to test. After the experiment, any developer knows that it’s not enough to deliver to testers a piece of software compiled and debugged. The software must have a good level of quality without violations and with a low level of complexity.

More significant is the cultural impact for testers. Now they know that a software component cannot be released until a routine has a too high probability to have a fault. It’s a complete different style of working, not more based on „someone feelings“ about the critical components.

Product and test managers have now the opportunity to require and to examine a quantitative report of the level of faultiness of the software under/after testing. It’s a different approach to evaluate in a more accurate way the quality of the products before the delivering.

Lessons Learnt

The first, in chronological order, lesson learned is about the necessity to have a significant baseline of data regarding the process before starting any improvement action. This was not our scenario because, in the analysis phase, the problems we identified were not confirmed by quantitative measures but only by the feelings and the impressions of the working teams. This lack did not facilitate our initial work in the PIE and did not facilitate the work of CEFRIEL in the phase of definition of metrics and models.

Working on software routines to reduce the number of violations may be a long and boring work but it does not present significant risks to create dangerous instabilities. Different is the scenario about complexity. Reducing the level of complexity of a routine often is not a very simple work and the risks to generate instabilities are concrete. The lesson we learnt about this point is that we have to perform this task gradually and in different steps.

Initially we planned to estimate the quality of the delivered software using measures of coverage (i.e. statement coverage or branch coverage) and to use these measures also as a criteria to determine when test activities may be considered concluded. While the experiment was running we understood that a measure of coverage represents only an indication about test effort (useful but not specifically requested in this experiment) but this measure cannot give us any significant information about the level of faultiness of the delivered software. Statistical models, following a different approach, try to estimate the potential residual faultiness of the software using, as baseline, static metrics. Consequently, if well tuned, they can give a more exhaustive information about the quality of a product.

Weak and strong points

A strong point is, without any doubt, the introduction of the concepts of metric as an important way to measure the quality of our process; now we have the rules and the tools to monitor in any moment the quality of our process and to perform any identified corrective action.

Another strong point is that any step of the new process improvement is fully based on the use of tools supporting automatically the capture of metrics, the detection of violations, and giving significant helps to reduce the complexity of software and to measure the level of coverage.

A significant weak point is the already mentioned lack of initial measures on the typologies of violations performed by the development team in the implementing phase.

The second weak point regards the „range of validity“ of the static and dynamic models. We cannot affirm that the models after the tuning phase performed in Elsag, can work fine on other software components and we cannot affirm that their goodness can be automatically confirmed forever. A continuous monitoring and, eventually, tuning may be necessary in the baseline project any time „external“ conditions (i.e. modification of the members of the development team) change. Identical consideration must be done when we’ll extend the models to different software components written by other developers with different style of programming.

Future actions

From the point of view of the future actions we'll proceed contemporary in two directions:

- We'll extend the results of the experiment to other software components with the target to improve and test, according to the new process, the complete set of software modules of a postal system. We plan to reach this goal in April 2001.
- We'll continue our collaboration with CEFRIEL. As mentioned before, metrics and models require frequent analysis, based on the captured feedback inside and outside the company, and tuning actions must be planned. We plan to analyze jointly with CEFRIEL metrics and models every six months.

This last point introduces the aspect of the reproducibility of the experiment. We assume that any technical result captured within this experiment is fully reproducible for any other software components developed in C\C++ language inside or outside our company. The points to be reviewed when the process defined in the current experiment has to migrate in a different environment are:

- the list of violations and their classification in classes according to the needs of another development team
- the static and the dynamic models must be tuned according to the different styles of programming of another development team.

References

[1] conte86a – S.D. Conte and H. E. Dunsmore and V. Y. Shen – „ Software Engineering Metrics and Models – Benjamin/ Cummings Publishing Company – 1986 – Menlo Park CA

[2] IB-A83098 – A. J. Perlis and F. Sayward and M. Shaw – „Software Metrics: Ananlysis and evaluation" – MIT Press – Cambridge, Mass. – 1981

[3] V. R. Basili. - Tutorial on Models and Metrics for Software Management and Engineering. - IEEE Computer Society, New York, 1980.

[4] V. R. Basili and E. E. Katz. - A formalization and categorization of software metrics. Technical report, Dept. Com. Sci., Univ. Maryland, College Park, 1986.

[5] S. Mohanty. - Models and measurements for quality assessment of software. ACM Computing Surveys, 11(3):251--275, September 1979.

[6] C. E. Walston and C. P. Felix. - A method of programming measurement and estimation. 16(1):54--73, 1977.

[7] J. P. Cavano and J. A. McCall. - A framework for the measurement of software quality. - Proc. Software Quality and Assurance Workshop, pages 133--139, San Diego, CA, Nov. 1978.

Appendix 1 - Author Profile

Traversa Mauro has 15 years of experience in Elsag in the development, testing and software configuration management areas. After the courses at the Electronic Department at the University of Genoa, Traversa was integrated in the Development and Research Department of Elsag in a team responsible for the implementation of a multi-processor and real-time proprietary operating system for internal applications. In this team he worked initially in the test area validating the programming interface of the kernel core of the operating system. After two years, he entered in the development team contributing to the implementation of software components for a new major release of the operating system. It's in this context that he started to be involved in software configuration management thematic and in 1994 he became the configuration manager in a critical and strategic join venture with a californian company for the implementation of a distributed system of character recognition. When the pilot project was concluded, Elsag formed a stable and independent team for software configuration management and Mauro Traversa assumed the responsibility of this team, actually working to manage any software component of a postal system. This team, initially dedicated to software configuration management activities, extended its competencies to offer to the development area a centralised service for installation procedures of software components.

Mauro Traversa has the responsibility of the best practice described in this article and he is actually involved in an Esprit project whose goal is the implementation of a tool for the automatic generation of test cases and for threats detection.

Appendix 2 - Company Profile

Elsag S.p.A established on November 1st 1998 was formerly a Division of Elsag Bailey, a Finmeccanica Company, one of the largest Italian industrial groups.
With revenues of about euro 360m in 1998, euro 410m in 1999, and more than 2,450 employees, Elsag S.p.A. group is now one of Italy's more important suppliers of IT solutions and services.

Elsag operates in the following areas.

- Postal automation, including:
  - postal mechanization,
  - electronic mail,
  - computerized management and control of postal service quality.
- Automation of image reading and recognition, with:
  - the reading and automatic processing of documents,
  - satellite and aerial survey,
  - object tracking and identification.
- Automation and supervision of various services including:
  - Energy Transmission and Distribution Network Control Systems,
  - on board ships and off-shore platforms Automation Systems,
  - fluid transport and distribution networks, and environment protection (monitoring of territory) Control Systems,
  - Integrated Management Systems for large buildings and structures.
  - Satellite communications,
  - Bank automation services.

Elsag Spa has been holding an ISO9001 certification since 1993.

Appendix 3 - Applied metrics and evaluation of the models

Metrics used in the models

Listed below, there is a short description of any software metric applied in the static and in the dynamic models.

LOC: lines of code: any line in the source that is not a comment or a blank line; therefore it includes standalone braces and parentheses on a single line;

eLOC: essential lines of code. Lines of code not including blank lines, comment lines and isolated braces or parenthesis lines;

Lines: total number of code lines, no matter what they contain;

Comments: lines containing comments;

FP: number of parameters of the function;

V(G): Cyclomatic complexity;

eV(G): essential Cyclomatic complexity;

CO: Comparison operators (<, >, = =, etc.);

LFC: Logic Flow Complexity;

OC: Operational complexity, weighted sum of present operations;

NEST: Maximum number of nesting levels of if-then-else control structures;

n1: Halstead number of unique operators

N: Halstead program length, calculated as the sum between N1 (Halstead number of total operators) and N2 (Halstead number of total operands);

n: Halstead program vocabulary, calculated as the sum between n1 (Halstead number of unique operators) and n2 (Halstead number of program operands);

V: Halstead program volume, calculated as $V=N * (\log_2 n)$;

D: Halstead difficulty, calculated as $D=(n1/n2)*(N2/n2)$;
E: Halstead effort, calculated as $E=D*V$;

BRANCH: number of forced exits from control structures by means of *goto*, *exit*, and *break* (also inside a *switch*) statements;

OAC: Operation argument complexity, weighted sum of arguments of each operation of a module;
ANION: Adjusted number of input/output nodes, calculated as the sum between number of entry nodes in a module and number of exit nodes from it, adjusted to behave intelligently where redundant "return" statements exist.
CONTROL: number of control statements;

EXEC: number of executable statements;

NSTAT: Number of statements (equals to CONTROL + EXEC);
QCP_MAINT: Quality criteria profile – Maintainability. Linear combination of other static metrics calculated as *MAINTAINABILITY = 3 * N + NSTAT + NEST + 2 * V(G) + Number of Branching Nodes*;

Evaluation of the models

We used R^2, *the goodness of fit*, as statistic coefficient to evaluate the experimental results obtained with the tecniques of Logistic Regression on static and dynamic models.

This statistic is not to be confused with least-square regression R^2 — they are built upon different formulae, even though they both range between 0 and 1 and are similar from an intuitive perspective. This statistic may be interpreted as the proportion of uncertainty in the dependent variable explained by the model. The higher R^2, the higher the effect of the model's explanatory variables, the more accurate the model. However, as opposed to the R^2 of least-square regression, high R^2s are rare for logistic regression. For this reason, the reader should not interpret logistic regression R^2s using the usual heuristics for least-square regression R^2s.

For the static model we obtained $R^2 = 0.5176$ which is quite good in the context of the Logistic Regression.
For the dynamic model $R^2 = 0.4251$.

Constructing testcases from derived requirements

B. Hindel
3Soft GmbH, Erlangen
U. Hehn
3Soft GmbH, Erlangen

Abstract

The choice of strategy to select the set of test cases which checks the maximum number of requirements and software paths is an optimization problem. Our approach is to approximate the optimal solution by using a combined strategy of module, integration, system, and acceptance tests. We observe how the requirements described in the module specification have been derived from the system specification.

Further criteria can be produced from the use profile and risk analysis. This will indicate which aspects should be tested at which test level. We expect test costs to be significantly reduced through the use of such weighting factors, as less important aspects will be tested less intensively.

Introduction

Test execution cannot prove that software is free from defects - for non-trivial software there is no test strategy that guarantees 100% coverage of requirements or paths using testcases. This presents an optimisation problem: to choose a test strategy or a combination of several strategies such that as many requirements or paths of the software as possible can be checked using testcases.

Industrial practice today shows that, in testing, strategy is rarely used. On the developer level, in the most progressive cases, the determined level of branch coverage is reached by testcases. A system tester is satisfied with a determined figure for remaining defects of a reasonable test suite. There is no known approach by which, an optimal combination of strategies between developer tests (module tests) and system tests to be reached, is looked for.

With our paper we wish to show how different test strategies for module tests, integration tests, system tests and acceptance tests can be profitably combined.

For the combination it should be taken into account that the tests on different levels can be carried out with the help of expected values that come from the various development documents. So, for example, a module test can be tested against a module specification and a system test against the system specification. For a sensible combination of module and system tests it should be noted how the written requirements from the system specification are developed in the module specification. This means that one is interested in the derivation of requirements over the various documents of a software development.

With the help of these "derived requirements" a requirements coverage over several test levels can be formulated and can be interpreted as an optimisation criterion for the test process. Further criteria based on usage profiles, risk analysis, complexity metrics and development histories can be found.

From requirements to test cases

Testing should always be against expected values or expected scenarios. These are based on the requirements that are specified in various documents during the course of the software development [1]. In the example shown here, we specify a software development process that consists of the following phases and the named known documents below (not complete). Table 1 shows which document is developed in which stage.

Phases	Documents
Requirements analysis	Requirements specification (RS)
	Acceptance testcases (AT)
Design	System design (SD)
	System testcases (ST)
	Integration testcases (IT)
	Module design (MD)
	Module testcases (MT)
Implementation	Source code (SC)
Module test	Module test report (MTR)
Integration test	Integration test report (ITR)
System test	System test report (STR)
Acceptance test	Acceptance test report (ATR)

Table 1

Derived requirements

A reference to a requirement that appears in a document and that is specified in an earlier document is a derived requirement. It follows in content the original requirement but includes in addition a higher level of detail in comparison to the former.

From a requirement several derived requirements can appear in the next document level. Likewise, a design decision can lead to several implementation parts in different modules. We wish to speak about implementation parts and derived requirements in a generalised way.

Testcases are ideally based on requirements i.e. testcases are likewise derived from requirements.

With these concepts agreed, consider now a "derivation tree" for requirements. The "derivation tree" of requirements spreads over the documents that are developed in a software project. The leaves of the tree are always either testcases or implementation parts.

Figure 1 shows such a derivation tree for requirements in our example project (see table 1).

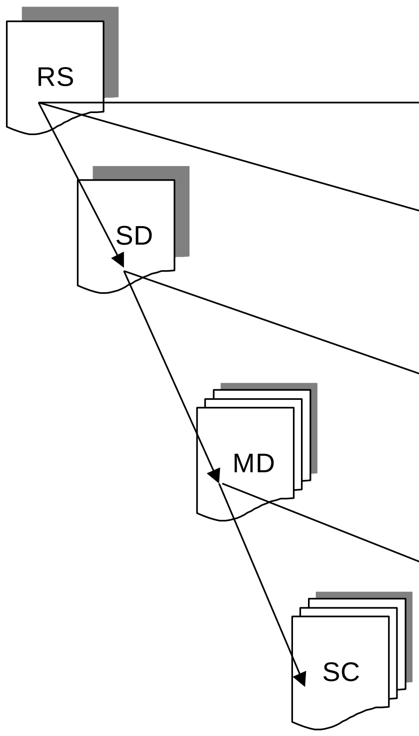


Fig. HIHE 1

Requirements keys

Each requirement is identified by a unique key within a project. This requirement key is chosen so that it gives information about

- the software project to which the requirement belongs
- the area for which this requirement is applicable i.e. a requirement is connected with the software system or with a particular subsystem
- the reference to a particular property within this valid area
- the ordering of the requirement within the requirements hierarchy

The requirement key is newly given first of all for an original requirement regardless of in which document this requirement is first defined. Derived requirements have the original requirement key in a modified form.

In our example, the requirement key starts with the character string "REQ". Then follows an unambiguous abbreviation for the project e.g. "EuroSPI2000". In the case that the requirement is only valid for one particular subsystem, in the same way an abbreviation for this subsystem can be appended. An unambiguous description from the theme of the requirement and including about 10 - 20 characters can follow.The individual parts of the requirement key are separated by an underscore (_).

Derived requirements are identified by the so-called point notaton i.e. at the end of the requirements key a point and a natural number are appended. In this way, as many derivations as desired can be developed from one requirement. For each further derivation step a new point and a new natural number are appended.

The construction of a requirement key as described above can be generally speified using Backus-Naur-Form as follows:

```
<ReqKey> ::= "REQ"   + "_"
                     + <project> + "_"
                     + [ <subsystem> + "_" ]
                     + <description>
                     + { "." + <number> }
```

Examples of valid requirements keys:

`REQ_EuroSPI2000_Deadline`	... original requirement
`REQ_EuroSPI2000_Deadline.1`	... 1st derived req. from derivation level 1
`REQ_EuroSPI2000_Deadline.2`	... 2nd derived req. from derivation level 1
`REQ_EuroSPI2000_Deadline.1.1`	... 1st derived req. from derivation level 2
`REQ_EuroSPI2000_Deadline.1.2`	... 2nd derived req. from derivation level 2

Test case matrix

Finally, the derivation trees for all requirements are transferred to a requirements matrix that has the original requirements shown against the project documents in a grid.

Table 2 shows such a matrix for the requirements R1 - R4.

Documents **Requirements**	R1	R2	R3	R4
Requirement specification	X	X	X	X
System design	X	X		
Module design	X			
Source Code	X			
Module testcases	X			
Integration testcases	X	X		
System testcases	X	X	X	
Acceptance testcases	X	X	X	X

Table 2

It can now be stated for each project what coverage of the requirements matrix is needed. It could be stated that, for example, the requirements that only appear in the requirements specification and which are not further derived from, are to be tested only in the system and acceptance test. Or it could be stated that requirements that appear in the module design must always be tested in the module test.

The lower part of the requirements matrix that contains the documents for the testcases can also be regarded as a "testcase matrix". When a trace can be carried out through the requirement derivations without gaps, this testcase matrix can be used to optimise all of the tests. In such a case it is very easy to state that each requirement must be tested at least once and possibly critical requirements at several levels.

Testing of requirements which own derived requirements

So far we have considered requirements and the derivation of requirements from the high level requirements specification to the low level source code. If we plan to test those requirements on different levels of abstraction we have to be aware of the fact that an original requirement may look similar but will not be identical to its derived requirements: The environment changes when the abtraction level changes. This means the situation is somewhat more complicated than first implied.

Consider the following situation:

- A requirement REQ of the requirement specification level is extended to requirements REQ_DERIVED.1, REQ_ DERIVED.2, ..., REQ_ DERIVED.n of the module design.
- The derived requirements are checked successful against the module implementation.
- Even when all derived requirements are fulfilled by the corresponding module implementations there is no guarantee that the original requirement is fulfilled at system level. For example, by mistake a module which is tested successfully may be not used at all in the system, or it might be used indeed, but incorrectly.
- Conclusion:
  The successful implementation of a derived requirement - proven by testing against the requirement - is not sufficient for the successful implementation of the original requirement.

What is the solution to the described problem? If we can guarantee that

1. the derived requirement REQ_DERIVED.x is fulfilled (at its lower level) – proven by testing it -
2. and the implementation of the derived requirement REQ_DERIVED.x integrates correctly into the implementation of the original requirement REQ (at its higher level) ,

then the original requirement REQ is fulfilled too.

If the system is skillfully designed, the check of the derived requirements should be more effective than the check of the original requirements, and the check of the correct integration of the derived requirement's implementation should be very simple. In sum by that method the

- total effort of testing will usually be lower as the test of the original requirements directly
- whilst guaranteeing a better coverage by testcases and therefore a better quality of the test itself.

Optimisation by test case selection

The complete requirements trace described above is not carried through to the testcases in every project because of costs. Often we must decide, which area of the developed software must be intensively tested and which not. In the following we wish to show briefly some selection criteria that help to choose the areas to test.

Usage profile

Frequently, the functionality of software is not equally used. One speaks of the 80-20 rule that states that 20% of functions in software are 80% used while 80% of functions are only 20% used.

In testing, our first objective must be to test the 20% of functions that are very frequently used. For this we construct a usage profile i.e. starting from the user functions we pursue the subsystems, modules or source code functions that have a higher probability of being called during a typical run of the software. These application functions, subsystems, modules or source code functions should be tested first.

A complete requirements trace aids the deployment of this strategy.

Risk analysis

Safety critical software does not need the most frequently executed parts to be tested, rather each area of the software, in which a defect can lead to serious damage. In this case, risk analysis is carried out with the question of which incomplete requirement leads to which damage. The requirements with the highest damage potential are tested first.

Defect probability

A defect rarely arises alone. Often judgement or understanding problems during the development lead to errors. One can show, with the aid of software metrics for software complexity and for software development histories, that defects appear most often in complex modules or in modules with considerable development histories.

Complexity

The complexity of a module can be found using several measures:
- the length of the module (lines of code)
- cyclomatic complexity (the number of branches)
- lexical complexity (the number of keywords e.g. user defined variables)
- levels of nesting
- size of the parameter list

One can give many further complexity measures that can help to determine modules that are especially prone to defects. If all modules cannot be tested then one can start with the modules that are, in accordance with the complexity measures, lying in the "top ten".

Development history

If in a design review there is a lot of re-work, then perhaps this shows a problem of understanding and the module should undergo intensive testing. Further indications of possible defects from the development history of a module are the number of participating teams, co-workers or subdeliverers, the level of re-use or the frequency of changes in the requirements.

Conclusion

The considerations shown were employed in projects in which safety critical software was developed. In one, for the programmming of X-ray machines (X-ray dose calculations for radiation therapy, aperture control etc.) a regression test was carried out that adopted the aforementioned optimisation. In this case, at least an 80% branch coverage was demanded of the developer executed tests prior to release. Of the second case, from the certifier FDA (Food and Drug Administration, USA), a demand for complete requirements documentation without gaps was made. This laid the basis for a general testcase matrix for module test, integration test, and system test. For the selection of tested modules for integration test a complexity measure, which was adopted to the programming rules, was used.

In another, the ideas shown were put to the test during the software development for motor vehicle cockpits. In this case, a graphic library for the LCD-cockpits for future motor vehicles was developed. This was an object-oriented development. The requirements were specified using use cases and were derived over the various UML diagramming techniques leading to testcases.

Up to now, the applied optimisation strategy has been reasonably convincing. either projects audits by the certifier nor subsequent usage have revealed any serious defects that remained unknown during the application of the test strategy.

References

[1] M. Roper: „Software Testing”, McGraw-Hill, London, 1994

[2] R. Pressman: „Software Engineering, A practitioner‘s approach”, McGraw-Hill, 1992

[3] Customer’s project performed by 3SOFT

[4] Customer’s project performed by 3SOFT

[5] Customer’s project performed by 3SOFT

WHEN
Release Decision Metrics

Niels Bruun Svendsen
& John A. Fodeh
B-K Medical A/S
Denmark

Introduction

When developing systems and software the inevitable management question is: "When is the system ready for release?". On the bottom line, the answer to when to release a new product for production and sales is a matter of being able to determine the balance between time-to-market and quality. Ship now and win market shares or postpone the release for a higher quality product? In order to estimate the economical impact of releasing a product, the release decision must be based on quantitative arguments and consequences.
This paper will share the experiences gained and the lessons learned from introducing metric based release decision support.

For setting up a metrics program, the process described in the CMU/SEI handbook "Goal-Driven Software Measurement" was applied. Apart from leading to a well-defined set of metrics, the impact on the organisation was remarkable. The metrics program resulted in a „Release Form“, i.e. a data sheet containing a set of metrics collected during the system test phase together with other relevant information needed for assessing the product's readiness for release.
A number of metrics included in the developed Release Form have been applied in multiple releases and the results evaluated after each release. This paper will highlight the benefits found as well as the problems encountered. Furthermore, it will put emphasis on the experienced effect of introducing and working with these metrics, that has been seen in the organisation.

The work has been supported by the European Commission and this paper is part of the final dissemination of the ESSI – Process Improvement Experiment (PIE) project 27498 – WHEN. The PIE has three objectives, of which two are related to release decision support, the topic of this paper.

Company Context

B-K Medical develops, produces and markets ultrasound systems for medical diagnostic imaging. The systems are sold throughout the world with the major markets being Europe, USA and Asia. B-K Medical has 250 employees with 170 located in Denmark. The development department consists of 60 employees where 20 are involved in software development. B-K Medical is ISO 9001 certified and most of the products have FDA market clearance and conform to the CE-Medical Device Directive. Therefore, internal as well as external audits are performed accordingly. No formal assessment against a model has been performed. Nevertheless, an informal self-assessment using the BootCheck tool from ESI has been performed. This assessment gave maturity ratings between 2.5 and 3.25, indicating some areas in need of improvement to get to the Defined (3) level, and a general lack of metrics as required in the Managed (4) level.

Project Initiation

The work on release decision metrics was initiated by the management group. It has its basis in the release meetings held within the management group, where the decision is made whether to release a new product or not. This decision was found to be based too much on subjective input and too little on objective data. Therefore, a goal was set to improve the basis for the release decision and thereby give objective support on when to release. A brainstorm on needed data and information to give an improved support for the release decision was held with the management group. The brainstorm was used as input for the work to be done and to get the full attention of the management group on the fact that the improvement work was started and that it was their problems and wishes being addressed.
In order to measure the impact of the improvement, a questionnaire was designed based on the output of the management brainstorm. The questionnaire was applied after each of three releases, with one release being before the improvement work was started, one being half way through and one being at the end of the improvement work.
The questionnaire consists of 7 questions to be rated in 1 of 5 levels. The questions are:

1. How would you characterise the basis for release decision in general?
2. How was the remaining known errors and their consequences presented?
3. How was the presentation of how much that had been tested?
4. How was the presentation of how thorough the user evaluation was?
5. How was the estimate on remaining unknown errors?
6. How was the estimate on remaining unknown safety errors?
7. How was the post-release plan presented?

The five rating levels are:
1. ___ Non existent
2. ___ Weak (Very subjective)
3. ___ Fair (Subjective, but well argumented)
4. ___ Very Good (Mainly objective)
5. ___ That’s how to do it (Objective, based on solid data)

The initial average score was 2.4, indicating that the basis for release decision was all together subjective. The results obtained during the WHEN project can be seen in The Results section.

Setting up a Metrics Program

At the time of the WHEN project definition a number of metrics were identified. In addition, the brainstorm with the management group gave input to additional measurements that could be applied. It was decided to test if the planned metrics and techniques were optimal for supporting the objectives of the experiment. The „Goal-Driven Software Measurement“ method was selected to set up the relation between the objectives and the measurements to be performed. This handbook, developed by SEI (Software Engineering Institute) at Carnegie Mellon University, ref. [2], delivers a formalised and structured method for decomposing the Business Goals of the PIE into a set of metrics and clarifying the dependencies between the metrics as well as the actions needed for collecting them.
The method builds on the GQM (Goal Question Metric) method by Basili and Rombach, and extends the GQM with a phase that guides the user from Business Goals through Sub-Goals to Measurement Goals. In overview, the method can be illustrated as shown in Figure NBS-JAF 1.

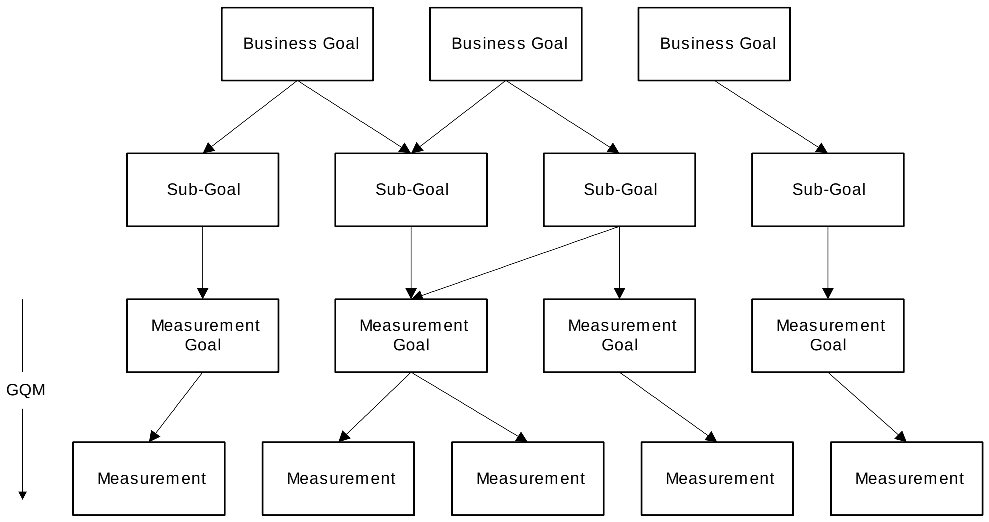


Figure NBS-JAF 1: Goal-Driven Software Measurement method

The work on Goal-Driven Software Measurement was conducted as a series of workshops involving the newly formed system test group and an external mentor. The system test group consisted of a senior test manager, a senior SW engineer and two newly employed test managers. The first step on the way from Business goals to Sub-Goal was to ask questions concerning the process involved. As shown in Figure NBS-JAF 2, this raised another question, i.e. which process are we talking about? It was found that despite the fact that work is performed according to an ISO9000 certified quality system, the process definitions were either lacking or not detailed enough and the terms used were not defined. The „Goal-Driven Software Measurement“ handbook uses a concept called „Mental Models“. Mental Models are the perception of procedures, processes and practices in the mind of the user. Models like that can work when only one person is using the model, although the model has a tendency to change according to the current situation. The problem occurs when more people are involved and only Mental Models exist, because there is at least one Mental Model per person involved.

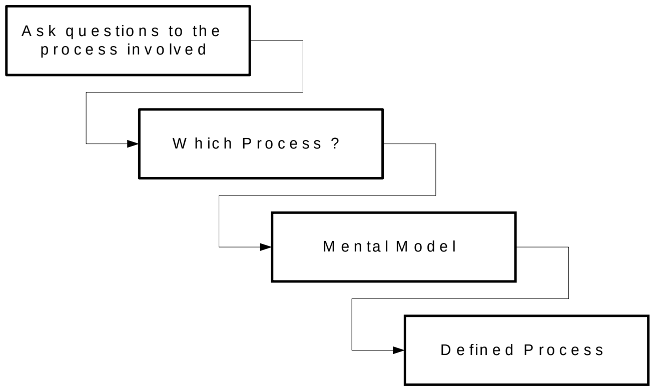


Figure NBS-JAF 2: Which process?

Getting the individual Mental Models aligned and written down in process definitions took quite some time. However, the discussions afterwards could be aimed at continuing the Goal-Driven Software Measurement process, instead of discussing the proper use of terms and which sub-processes existed.
Having reached the point where a number of Sub-Goals were defined we were ready to apply the GQ(i)M part of the process. The (i) part stands for indicator and is an addition to the GQM that we found valuable. The idea is to make sketches of the desired presentation of the measurement results. It

makes the measurements more real and „alive“ and generates a number of additional discussions and ideas. Examples of indicators can be seen in Figure NBS-JAF 3.
Several measurements were defined using the Goal-Driven Software Measurement method. A number of these were selected as our release decision metrics. It was noticeable that some Sub-Goals did not directly result in measurements, but rather pointed out the need for templates, checklists etc.

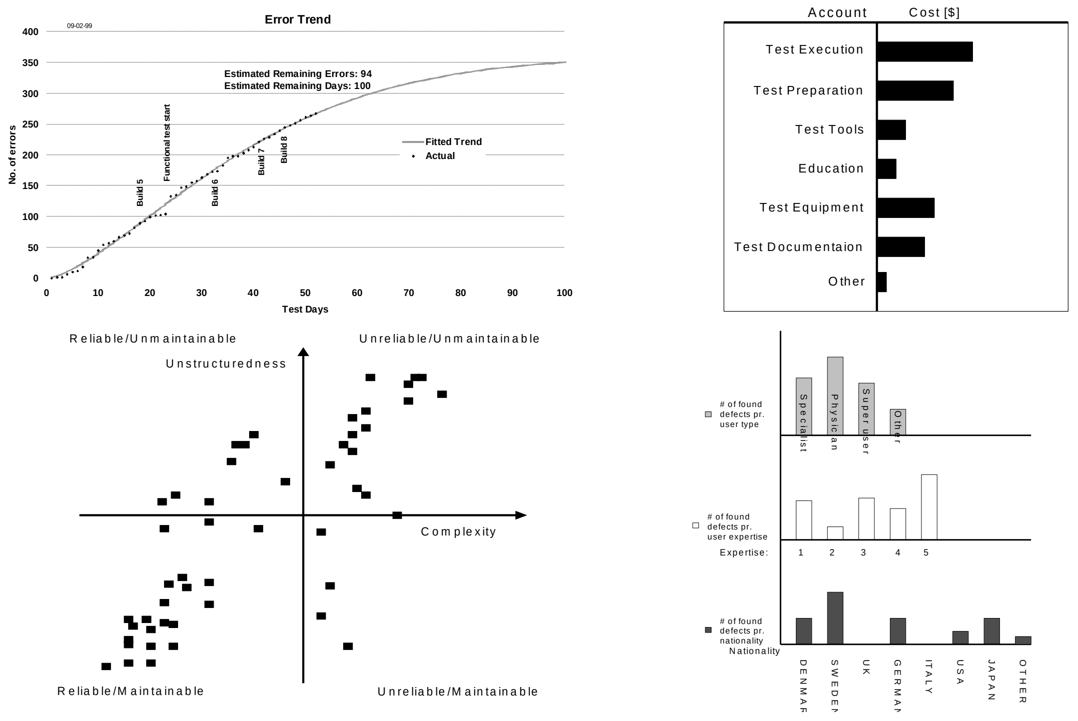


Figure NBS-JAF 3: Indicators

The final step was to prepare a plan that addressed the identified actions needed for both implementing the measures and completing the templates and checklists.
This plan set the framework for the WHEN PIE activities and established a reference for further improvements of the processes.

The Release Decision Metrics

The selected release decision metrics can be grouped into the following 4 main groups:

- System Status
  - The number of known remaining errors and their consequences
  - System stability data
- Estimate on remaining errors
  - Altogether and sorted according to severity
- Test Coverage
  - Test execution coverage
  - User evaluation coverage
  - Code coverage
- Post release plan

Based on these, a Release Template was developed.

The Release Template

In the following, the developed Release Template is shown. The release template is a data-sheet containing the gathered metrics regarding the state of the system to be released. The data sheet is

usually delivered to the management group some hours, or the day before the actual release meeting, so the contents of the sheet can be studied in advance.
In the template, the actual value of each metric is shown together with a target value and a reference value (i.e. the actual value from a former release of the same or another product). At this time, targets have only been set for a few of the metrics. It is planned to add further target values as we obtain the data to base the target values on.
In addition to the metrics table, the release template includes two charts. One showing the stability of the system for each of the builds during the system test phase (Figure NBS-JAF 5) and the other showing the error trend based on the error detection rate (Figure NBS-JAF 6). The error trend has been one of the key metrics introduced and will be described further in the Error Trending section.

The test coverage data include the information concerning the progress and completeness of the testing. A low value reveals insufficient testing effort and the risk of potential latent defects. It is planned to extend this section with code coverage data for quantifying the portion of the code that is exercised by testing, thereby showing the thoroughness of the applied testing techniques.

The system stability section delivers vital information about the reliability of the software. The data shows the mean number of operations between failures, equivalent to the widely spread Mean Time Between Failures (MTBF) metric. This section refers to the stability chart giving a graphical presentation of the mean number of random operations between failures, as a function of the build number. The chart contains two limits; the lower limit is the entry criterion for system testing, while the higher limit is release stability criterion. In this way, it is straightforward to confirm that the system's stability is adequate for release.

Test system status section contains statistics regarding the problems reported during the system-testing phase. The total number of problems reported is shown and categorised in closed (fixed and verified) and open problems. The open problems are sorted according to their severity. These data deliver a snap shot of the system state at release time, making it possible to take into account the risk and consequence of releasing the system. E.g. if the data reveals a large number of open high-severity or non-verified problems, then it clearly shows that releasing the system at this moment is high-risk decision.

User feedback during the development is undoubtedly of major importance. The user evaluation section presents relevant data collected during the user evaluation activities. At this time, this section only contains a summary of the raised problem reports and their classification. It is planned to extend this section with information about covered applications, user types, countries, etc.

The post release plan section contains an overview of the activities to be performed after the release of the product together with the responsibilities, schedules and the date for the subsequent release. The post release plan sends a clear signal that the project is not ended with the release of the product. This helps preventing management from allocating all resources to new projects just after release. Instead, efficient planning in the transition phase between projects can be made.

System XXXX, Version: YYYY			
	Value	**Target**	**Reference**
Test Coverage			
Percentage of executed vs. planned test scripts		100%	
System Stability (see Fig.1)			
Mean number of operations between failures		80.000	
System Status			
Number of raised problem reports			
Number of closed problems			
Number of problem corrections to be verified			
Number of known problems in the Build XX			
– Number of **safety** errors		0	
– Number of **functional** problems			
– Number of **discrepancies**			
– Number of **change requests**			
Error Trend (see Fig.2)			
Estimated **total** number of problems after release (first ½ year)		<5	
Estimated number of **safety** errors			
User Evaluation			
Number of raised problem reports			
– Number of **safety** errors			
– Number of **functional** problems			
– Number of **discrepancies**			
– Number of **change requests**			
Post Release Plan			
The following activities/issues will be addressed to next release in the following priority: 1. ... 2. ...			
Allocated resources – Software: ... – Hardware: ... – System Test: ...			
Planned date of 2'end release			

	Test Personal
Test Manager	Initials
Test Engineers	Initials
Test Technicians	Initials

Figure NBS-JAF 4: Release Template

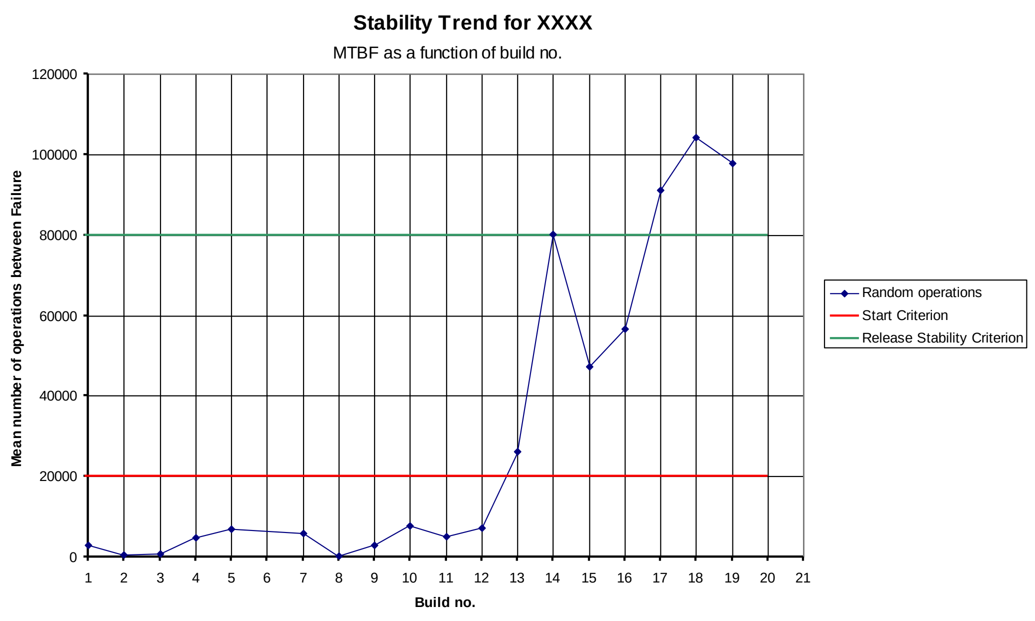


Figure NBS-JAF 5: Stability Trend

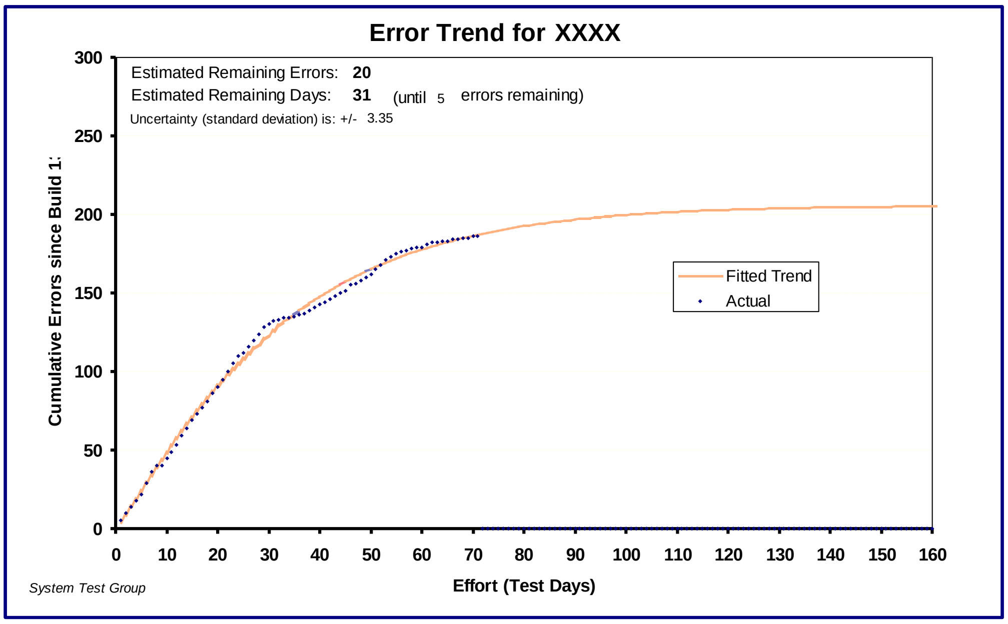


Figure NBS-JAF 6: Error Trend

Error Trending

For estimating the number of remaining unknown errors, error trending is used. Error trending is based on the graph showing the accumulated number of reported errors (y-axis) as a function of test effort (x-axis), as shown in Figure NBS-JAF 6. The test effort is expressed in terms of test days. A test day is the effort equivalent to a typical (8 hours) work day of a single tester.

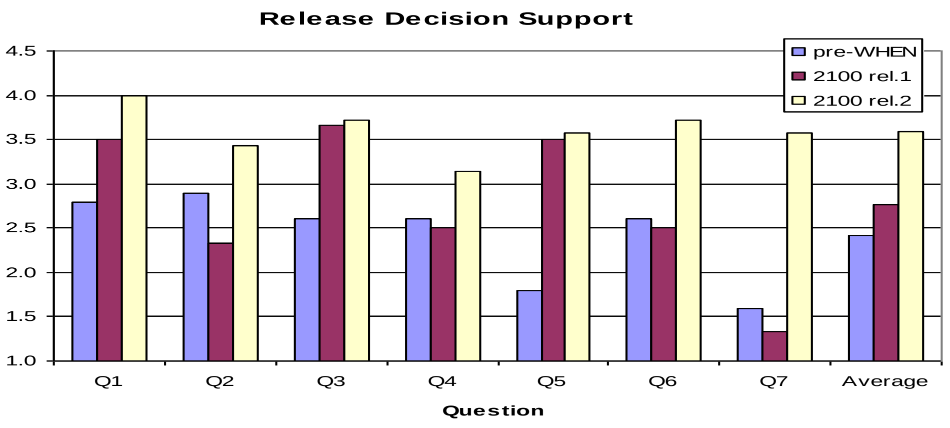
The dots in the graph represent the reported errors, while the line going through the dots is the best-fit line (mathematical least square) based on the Weibull function. This line is extrapolated, providing a predictive evolution of the error finding rate.

As noticeable, the graph is S-shaped and can be divided into three sections; the first is the section with the slight slope at the beginning, the second is the mid-section with the linear-like slope, the third is

the section where the graph flattens out. This S-shape is found to correlate with empirical data from software projects. At system test initiation, the error finding rate is low (as the functionality of the software is often restricted to few areas). The error finding rate increases with the addition of new functionality and the introduction of new errors during the correction of already found errors. Entering the third section, the error finding rate begins to decrease, as it becomes harder to find new errors. Ultimately, the graph flattens. Finding further errors at this stage require huge test effort and shows that the software is possibly ready for release (or that the limitation of the applied testing technique has been reached).
More details on the error trending can be found in ref. [1] and the results obtained by using it are discussed in The Results section.

The Results

Measurable results have been obtained on the quality of the release decision support. Equally, on the precision of the Error trend based estimate of the number of remaining unknown errors. The quality of the release decision support has been measured by means of the previously mentioned questionnaire. Figure NBS-JAF 7 shows how the rating of each of the 7 questions has evolved through the 3 releases. Included is also an average of the 7 questions for each of the 3 releases. It is seen that the average score has increased from 2.4 to 3.6. With the level definitions in mind, this means that the basis for release decision has been moved from all together subjective to mainly objective.



Q1: How would you characterize the basis for release decision in general
Q2: How was the remaining known errors and their consequences presented
Q3: How was the presentation of how much that had been tested
Q4: How was the presentation of how thorough the user evaluation was
Q5: How was the estimate on remaining unknown errors
Q6: How was the estimate on remaining unknown safety errors
Q7: How was a post-release plan presented (regarding further testing and further releases)

Figure NBS-JAF 7: Result of Release Decision Questionnaire

One of the major improvements is the estimate on remaining unknown errors. This estimate is based on the error trend. The results of the error trend based estimates compared with actual number of errors found can be seen in the table below. What we conclude from this, is that the error trend based estimate is an optimistic estimate. It is not high precision, but it is fairly consistent and far more realistic than a subjective estimate. The experience is that the error trend based estimate is nearly always received as being high, i.e. „Do we really have that many errors left“. In that case, it is important to notice that so far the estimate always has been too low.

	Trend based estimate	Reported after release[1]

Pre-WHEN release	5	15
2100 release 1	17	35
2100 release 2	31	35[2]

[1]All reports counts, including change requests.
[2]Only 3 months data available. The other results are based on 6 months.

Lessons Learned

Metrics are valuable in planning and decision making
Clearly, there is a substantial pressure to maximise the profit by releasing early. On the other hand, the economical losses of releasing a poor-quality product as well as the damages in goodwill and reputation may inflect irreversible damage. In the lack of metrics to support the release decision, the state of the system to be released is vague, often resulting in an unnecessary delay of the release.
In this respect, the metrics used for supporting the release decision have shown their value. By giving the management group a more objective release decision basis, a higher degree of freedom in their decision has been obtained. A visible effect has been that management has decided not to delay releases in order to reduce the number of unknown defects at time of release, but to focus on a post-release plan to bring down the impact of post-release errors.

In the planning phase, the metrics have also shown their strength. The ability to give a qualified guess on the effort size of a system test project 9 months ahead, by use of the SW development time to system test time ratio, is convincing. During the system test, the error trend has given input to the planning of the remaining amount of test and needed resources for both testing and error correction.
Furthermore, metrics have also taken the role of a common reference. Especially the stability and error trends gave the common reference for discussion of system state, i.e. a simple graph gives the common basis for discussing system state, which is understood and accepted by top-management, project management, developers and QA staff.

Metrics demand maturity or the will to mature
Working with defining relevant metrics we soon discovered that there was a need for clear definitions of the processes to base the measures on. In other words, for the metrics to be relevant a certain level of maturity is required. We did not initially have that level of maturity but we used the work on metrics to trigger and drive the improvements of process definitions. We experienced major benefits from that work especially in terms of job motivation, as there is no longer any need for spending time on the general way of performing regular routines, instead more effort can be put into solving the specific task at hand. Moreover, when spending time on the process it is to improve it, instead on figuring what the process is.
As much of the work done was focused on the system test phase, the major impact has been seen in the system test group. The results obtained as well as the discussion generated during the PIE has helped greatly in forming a dynamic and committed group that considers metric-supported process improvement a vital part of the process.

Conclusions

Incorporating metrics into the development process delivers an effective tool for planning, monitoring, predicting and following up. In particular for finding the proper timing for release, possessing the right metrics has shown a tremendous value. These metrics provide insight into the state and quality of the software system making it possible to base the release decision on solid data and well-calculated risk rather than intuition and gut-feel.

The conclusion on the use of Goal-Driven Software Measurement to drive the definition of a metrics program is that it can be highly recommended. Although it involved far more work than initially anticipated, it was undoubtedly worthwhile. Looking back, it was a necessary step for bringing up the level of maturity to where measurements start to make sense. Starting out without the awareness of missing process definition etc., the Goal-Driven Software Measurement was a perfect trigger of the needed improvement actions. Especially in the system test group, the work completed with Goal-Driven Software Measurement had helped establishing a solid infrastructure consisting of well-defined, functional and efficient processes.

In relation to the release decision support, a clear and positive effect has been seen. The greatest positive effect has been seen for the error trend based estimation of number of remaining errors and for the post release plan. These improvements have also triggered an interest in other metrics based on available data. An example of this is the calculation of the general cost of delaying release and comparing that with the cost of field update of the SW. It showed that the cost of updating the SW on all scanners in the field, 6 months after release, equals the loss of delaying the release by only 10 days. The substantial cost of delaying the release shows the enormous pressure to release early and emphasises the importance of choosing the right release time, as the consequences of a "premature" release may be unrecoverable.

The developed release template will without doubt be used on future releases. It will be enhanced with code coverage and an improved user feedback section. It will also evolve towards defining release criteria by defining more target values.
In a broader sense, this work has helped establishing process improvement as a natural part of daily life in the development department.

References

[1] Niels Bruun Svendsen, Error Trending, Why and How, in: *Proceedings of the EuroSPI'99 Conference,* pp. 11.41-11.51, Series A – Pori School of Technology and Economics, No A 25, Pori 1999

[2] Robert E. Park, Wolfhart B- Goethert, William A. Florac, „Goal-Driven Software Measurement – A Guidebook“, Handbook CMU/SEI-96-HB-002

[3] Linda Rosenberg and Lawrence Hyatt, „Developing a Successful Metrics Program“, Software Assurance Technology Center (SATC), USA, 1997

[4] Stephen H. Kan, „Metrics and Models in Software Quality Engineering“, Addison-Wesley, USA

Session 8 - SPI and Measurements

Session Chair: Mads Christiansen, Delta

Enhancing the Measurement of Process Maturity and SPI effectiveness

John Elliott,
SEC, DERA UK

Abstract

This paper analyses and assesses the measurement strategies that are contained within process 'maturity' assessment methods, such as CMM and SPICE, and considers their potential extension to focus on the evaluation of the business effectiveness of SPI (i.e. to benefit both the developer and customer businesses). A common assumption is that formalised measurement procedures become more important (and necessary) when development processes are judged to be at a higher maturity level. Such procedures address project, or SPI, performance but they only partially address process effectiveness and efficiency. This paper argues for extended cost-effective measurement schemes to support many kinds of decision-making activities that apply throughout the evolution of SPI from low to high maturity; these schemes will enable the business value and payback of SPI to be continuously assessed. Furthermore, the paper proposes and outlines a measurement framework that will enable more continuous (i.e. representing a self-assessment' paradigm) business-led process 'maturity' assessment; this framework will operate across (and facilitate the integration of) business models, business-led SPI models and technically-led process maturity assessment models.

Quality Metrics and the Kano Model

Tor Stålhane
SINTEF Telecom and Informatics,
Trondheim, Norway

Introduction

This paper consists of three parts. The first part discusses the Kano quality model [1] and its consequences for the definition and collection of quality metrics and for software process improvement - SPI. The second part shows an example of the application of the Kano model in the assessment of the reusability of software components in the REBOOT project [2]. The last part of the paper is the conclusion, where we try to sum up what, in our opinion, should be the Kano model's consequences for the work on software quality metrics. In addition to the Kano model, this paper also introduces a plotting technique that can be used to analyze quality data with respect to this model.

The Kano model

The Kano model's quality concept

We will use the term quality as it is defined by ISO 9000, i.e. quality is the product's ability to satisfy the customers' stated and implied needs – in other words, the products' ability to please the customer. The Kano model for quality factors is simple. Basically it says that there exist three types of quality factors:

1. **The obvious**, which must be present in order to sell the product but will not give you any special credit.
2. **The required**, which is what the customer requires, and will be in your favor - the more the better.
3. **The surprise** – here used in the positive sense - which is neither required nor expected by the customer, but will give you a competitive edge. Some people call this type of factors the "wow factors" and quite rightly so.

ISO 9000 only consider the first two of these, where the obvious is included in what ISO 9000 calls implied needs. Surprise quality does not exist in the ISO world.

At a given point in time, any product or service characteristic will belong to one of the three types of quality factors listed above. Depending on its type, a factor's presence or absence will influence the customers' perception of the product or service as illustrated in the diagram below.

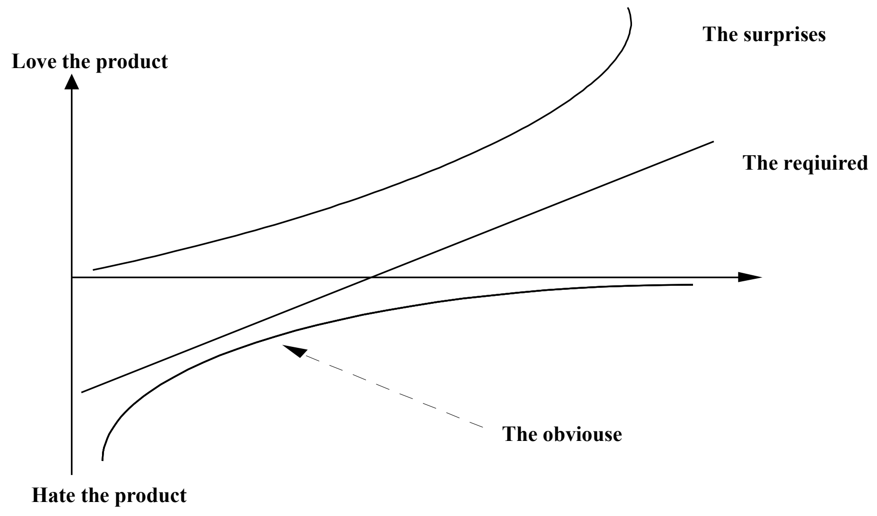


Figure 1 The influence of the three types of Kano quality factors

If we want to measure customer satisfaction or changes to this satisfaction, we see from the diagram that the type of quality factor in question will influence the type of responses we get. No change in an obvious quality factor can make the customers love the product and no lack of a surprise factor can make the customer hate it.

The Kano model and ISO 9126

Many people, especially those working as software developers, have problems with the Kano model, mainly because – as opposed to the ISO 9126 model - they can see no software-related terms there. The reason for this is simple – the Kano model is concerned with quality as seen form the customer, while the ISO 9126 model is a model that can be used to split a system's characteristics into a set of partly independent factors – called quality factors. The ISO 9126 model only describes a possible set of quality factors – the quality assessment must be done separately. As can be seen from the definition, quality is always a relationship between a product or service and one or more customers.

The Kano model is not concerned with whether a feature is important or advanced or not or is endorsed by one guru or another. The model is concerned with how the feature pleases the customer. Kano's message is that you need to put in at least one surprise factor into your system in order to compete in the market place. If you ignore this message you run the risk of building technical superb, but irrelevant products. As a result of this, you may be driven out by your more streetwise competitors.

Software Metrics and the Kano Model

Metrics collection

In the general case, it is possible to differentiate between the three types of quality factors if we ask our customers to rate the product in the absence and presence of these factors. We should observe the following:

- The obvious: Absence will result in negative reactions such as “do not like”, “will not use” etc. Presence, on the other hand, will not reverse the response but only move it to neutral.
- The required: Here, just as for the obvious characteristics, absence will result in negative reactions. Unlike the previous case, however, presence will create positive reactions such as “like the product” or “will use it”.
- The surprises: In this case, absence will create neutral reactions, while presence makes the customer give highly positive or even enthusiastic reactions to the system.

When we collect metrics pertaining to product quality, we need to keep the three above mentioned response patterns in mind. This means for instance that if we observe that an improvement in product characteristic C does not increase customer satisfaction, we cannot - and should not - infer from this that C is unnecessary. It might be an obvious quality factor. Removing it will then spell disaster for the product.

A similar case may exists for another characteristic D which none of our customers have requested. Instead of dropping it right away, we should check if it is a surprise quality factor. If we do not check this, we might lose an opportunity to make not just a good product but also a really outstanding one.

The three types of quality factors have different mechanisms and techniques used in order to discover them and in order to develop a product, which contains them.

- Surprise characteristics: These depend on application and market knowledge and insight. Important techniques that can be used to discover these characteristics are market surveys and QFD - Quality Function Deployment - see [4]. It is important to bear in mind that we can always check the customers’ reaction to a certain product attribute but we can ***not*** ask him to specify a surprise attribute.
- Required characteristics: These are the ones that the customers or the marketing department have required. The only challenge here is not to lose any of them during the development process. In addition to QFD, quality assurance and requirement traceability matrices can help us here.
- Obvious characteristics: These are internally generated requirements, stemming either from the project itself or from the marketing or quality department. Since they are already specified, the job of keeping them in place for the product belongs to the quality department. These quality characteristics are those that the ISO 9000 calls implied requirements.

It seems clear that it is important to know which of the three groups a certain product characteristic belongs to. There is only one way to find out - instead of just asking what happen if a certain characteristic is present, we also need to ask what will happen if the same characteristic is missing. One way of doing this is the method used in the reusability research done in the ESPRIT project REBOOT - see [3].

Metrics collection and the Kano model

The REBOOT example shows that the way the metrics are collected will influence our results. The REBOOT project used a questionnaire to decide the importance of each of a set of reusability factors. The main point in the reusability questionnaire concerning the Kano quality model was to ask the respondents to score both the presence and the absence of each characteristic. Thus, instead of asking them to rank each characteristic X on a scale of importance we asked the respondents to score each of the following events:

- X is not present
- X is present to some degree

- X is fully present.

We used the following schema for assigning scores:

- Strong positive influence: ++
- Some positive influence: +
- No influence: 0
- Some negative influence: -
- Strong negative influence: --

In the REBOOT project, we accumulated the number of uses of each influence score for each quality factor and then checked whether the number of positive or negative indicators was statistically significant. Our assumption was that if a respondent does not have any particular opinion, the probability of a positive or negative score should both be 0.5. Factors having one of the three types of quality factors identified by Kano should produce one of the patterns shown in the diagram below. The diagrams show the expected scores for "required" in the top left diagram, "surprise" in the top right diagram and "obvious" in the bottom left diagram.

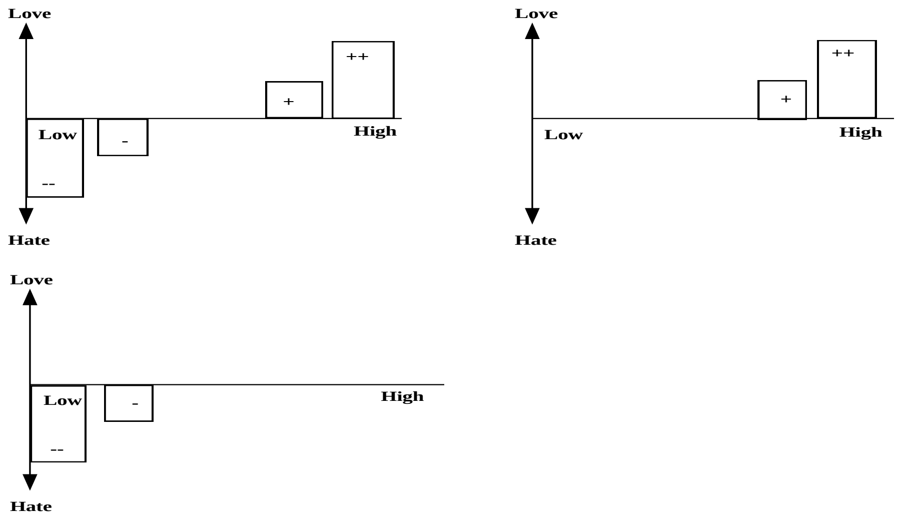


Figure 2: Scoring patters of the three Kano quality factors

Note that each factor - in addition to be a surprise factor, a required factor or an obvious factor - can be strong or weak. Strong factors contain a significant number of "++" or "--", while a weak factor only contains significant numbers of "+" or "-". For a discussion of the use of statistical significance and tests in this case, see [3].

The Lillestøl plot

The use of the three diagram types in figure 2 depends on being able to build statistical confidence intervals for the number of responses. This is, however, at odds with the basic idea of TQM, namely that everybody in a company shall be able to participate in the quality work and in the SPI of the company. In order to achieve this important goal, we need a simpler tool for selecting quality characteristics. The diagram shown in figure 3 is taken from [5] and is simple and intuitive to use. In order to include all responses, Lillestøl has also included that a factor can be assessed to be indifferent, i.e. the customer does not think that it contributes in neither a positive nor a negative way to his feelings concerning the product. The plot is used in the following way:

- We use the number of respondents who define a certain product characteristic as a surprise or a requirement respectively as the vertical or horizontal offset in the diagram.
- Each point in the diagram is drawn as a group of symbols using symbols describing the percentage of the respondents who define the factor as Obvious ("."), Required ("-"), Surprise ("+") or Indifferent ("/"). Thus, for instance, factor A has 60% surprise votes, indicated by six "+" in the circle. In this way, we can also include the two factors Obvious and Indifferent. We will use the following table as an example for the plotting technique:

Table 1 Example data

	Quality factor type			
Element	Surprise	Requirement	Obvious	Indifferent
A	26 – 64%	8 - 20%	3 - 8%	3 – 8%
B	3 – 8%	25 – 62%	10 – 25%	2 – 5%
C	4 – 10%	4 – 10%	30 – 75%	2 – 5%
D	3 – 8%	7 – 18%	1 – 2%	29 – 72%

Instead of statistical analyses, we can now select the most important quality factors from the diagram, based on a common understanding of the personnel involved. Note that both obvious and indifferent factors will be found close to the origo.

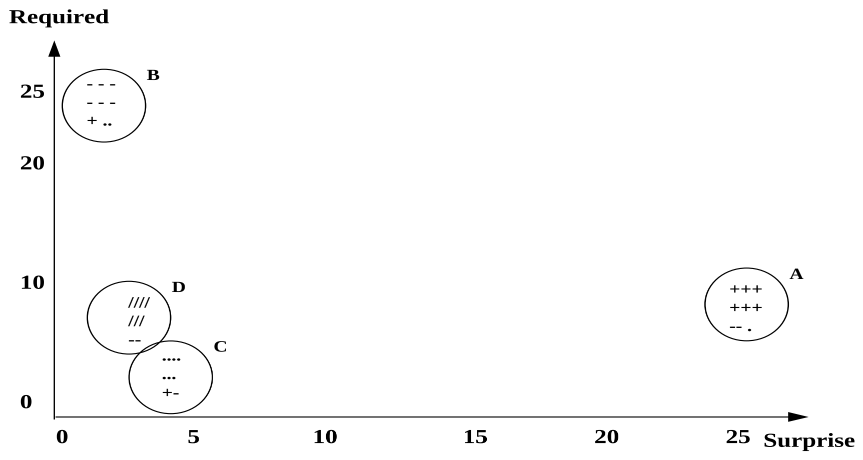


Figure 3 A Lillestøl plot for the data in table 1

The data in table 1 are chosen just to make a nice, self-explanatory diagram, and the plotted points are placed just where they should be in order to be surprises, requirements or obvious factors respectively. Responses from a real world experiment will not necessarily be so well ordered and will most likely be spread all over the diagram. It is, however, not difficult to interpret points in other parts of the diagram. If we leave out the "Indifferent" answers, figure 4 gives the general picture.

Since quality is a subjective characteristic, a factor or element that is a surprise to one person may be obvious to another. A typical example is the ISO 9126 factor "maintainability". To most customers it is an indifferent factor but it is a requirement or even a surprise to the persons doing the maintenance. This difference can be clearly seen in table 4 in the appendix where for instance "no errors after changes / updates" (question 31) is a requirement for 15 of the responders, while it is irrelevant for 10 of them and only three responders consider it a pleasant surprise. This is what we should expect when quality is subjective.

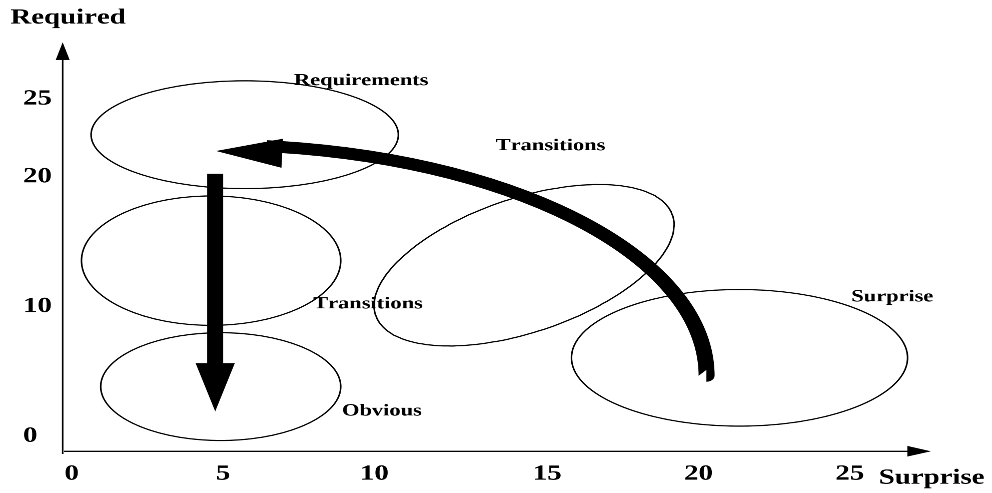


Figure 4 The transition of quality characteristics over time

It is important to watch those quality factors that fall in the two transition zones – the one between “Surprise” and “Requirements” and the one between “Requirements” and “Obvious”. Those that are moving from surprises to requirements offer us the opportunity to be in the forefront in satisfying our customers. Those that are moving from requirements towards the obvious are also important, since these are factors that are important in order to sell the product, even though our customers will stop asking for them. It is, however, important to bear in mind that factors may also move from requirement to indifferent. Missing this distinction may be a costly affaire. The number of “/” in the Lillestøl plot will tell us which factors that are obvious and which that are indifferent.

The Lillestøl plot has, however, two important drawbacks:

- It cannot be used to show characteristics that do not fit the like - dislike scheme. An example is the comment density in code to be reused. Here we instead need a frequency plot before we decide what is important for the customer.
- Since we are only registering the type of each quality characteristic, we cannot see whether it is strong - for instance “--, ++” - or weak - “-, +”.

Consequences

Consequences for software development

When discussing the Kano quality model with personnel involved in development projects, the reaction has often been that they do not want to implement functionality that the customer has not required. Extra functionality will cost extra time and money and the customer will often not appreciate it. Often to the contrary - unspecified functionality is often considered quite a pain in the....

Surprise qualities – the top-notch characteristic in the Kano model – are important because they will help the product to stand out among its competitors in the market place. With such characteristics, your software is interesting and rewarding to use, without them your product is just one more software package. This is not a new observation. The battle between the video systems VHS and Beta was won by VHS even though it had a lower picture quality. The customers, however, thought that the picture quality of VHS was good enough - required quality – and chose VHS because it had a much smaller cassette – surprise quality.

The examples are legion. The Windows operating system was – when it first entered the scene – in many ways inferior to its competitors, for instance Unix. The use of icons and the "point and click" method was, however, a surprise quality and even if most software development pro's snickered, the Windows operating system was and still is a commercial success.

Finding the wow-factors is difficult due to the dependency between the role of each quality factor and the customer or market. This was brought home to me quite forcefully during the REBOOT project. In this project it was decided that all personnel should use a new word processor. This decision was greeted with great enthusiasm by many of the project participants. The main reason for this was that the tool was WYSIWYG (What You See Is What You Get). On the other hand quite a lot of people though that the new word processor was a terrible idea. The reason was - OK, so you outguessed me - that the tool was WYSIWYG!

It is important to bear in mind that there are several other areas besides new or improved functions where we can add "surprise" quality factors. One important area where the software industry has a large potential for improvement and which the customers think is important is the service provided to the users [6]. In my opinion, this is an area where companies that provide software really can get a competitive edge. Other promising areas are user interface and ease of installation.

The most promising wow factor of them all - at least for the professional market - will be the availability of a reliability warranty. At the present, most software comes with a statement, which really says, "We, the system provider, will shoulder no responsibility at all for whatever happen to you when you use this software". The first company that dare take one or more steps away from this rather pathetic state might grab a large share of the professional market for software.

A quality factor does not belong to a certain category forever. It is a common experience - not limited to the area of software systems - that what was a surprise factor yesterday is a requirement today and will most likely be an obvious factor tomorrow. The search for ways to surprise and delight the customer will never end. Since surprise factors over time will move on to become requirements, there is always a window of opportunity for an innovative company where they can find a new way to please the customer. This window lasts *from* the point in time when the possibility to implement the factor is realized and *until* the factor is changed to a requirement or an obvious factor.

Consequences for SPI and QA

Traditionally, the QA department has focused on not "loosing" customer requirements and has developed mechanisms such as the requirements tracing matrix for this purpose. The SPI people, often working hand in hand with the QA department, have focused on improving the process currently used. In TQM as in many other SPI approaches, the focus has been on registering customer complaints and removing their causes. This leads to a continuously improved process, which gets increasingly better in satisfying the customers' *stated* requirements. The improvement process is, however, still reactive.

However, the surprise quality factors cannot be discovered or inserted in this way. Thus, the quality factor improvement process is not enough. In addition, we must have a quality factor search process, intended at discovering surprise factors. This will enable us to grab new markets and new customers and leads to a new way of looking at the development and improvement processes. If we apply the Kano model, another important consequence for the QA department is that missing obvious quality characteristics may carry a quite heavy penalty from the market. Thus, this is an area where we cannot afford to fail.

Traditional SPI will not give us any “surprise” factors. Thus, we need to move to a more proactive way of doing SPI. The difference between a traditional and a new SPI approach can be illustrated by using the following table of contrasts - taken from [7]:

Table 2 Evolution versus revolution in SPI

	Evolution			Revolution		
Quality view	Problem to solve - a necessity			Ideal function to be achieved – an opportunity		
Motive	Eliminate complaints			Elicit compliments		
Step	Inspect	SPC	QA	Strategic quality management		
Focus	Discover	Control	Avoid mistakes	Listen to the voice of the customers		
Method	Measure	Statistical techniques	Programs and systems	QFD	System approach	Robust design
Responsible	Inspector	Production	Production, development	Everyone		
Result	Earn right to sell Keep market shares			Command higher price Gain market share		

The most important differences are the ones in the rows marked “Focus” and “Responsibility. The shift of concepts is easily seen. In the “Focus” part we move from the rather defensive technique of control and avoiding mistakes to the more active “Listen to the voice of the customers”. In the “Responsible” part, we move from a situation where the inspectors are responsible for the product’s quality to a situation where quality is everybody’s responsibility. This will influence the way the organization thinks about both quality and SPI. Last, but not least - this cannot be achieved by mimicking some “best practice” model. We need to go out and understand and listen to our own customers and to our customers-to-be.

Software reuse and the Kano model

The research that resulted in the REBOOT reusability factor - characteristics model is documented in [3]. The results are summed up in the table in the appendix. If we combine the factors that are requirements into one group and those that are considered surprises into another group, a clear picture emerges:

- The requirements are concerned with internal relations - mostly related to what the component contains. Typical items are documentation, input part separated from output part and robust design.
- The surprises are mainly concerned with how the component behaves and how it was developed. Typical items are demos, test cases, version and variant history and QA procedures used during development.

The results in table 3 come from statistical analyses of the accumulated scores from the REBOOT questionnaires. If we instead use the Lillestøl plot – see table 4 - we get the diagram shown in figure 5. In some ways, this plot is easier and simpler to interpret and provides a more detailed picture. These detailed were lost due to accumulation of the individual scores in the REBOOT project. The plot shows us that:

- The factors inside the rectangle - interpreted as surprise factors in the REBOOT project - are really in the transit area. They are still considered to be surprise factors, but may soon be requirements.

- The requirements inside the oval have been - and to some extent still are - requirements but are probably on their way to become obvious factors.

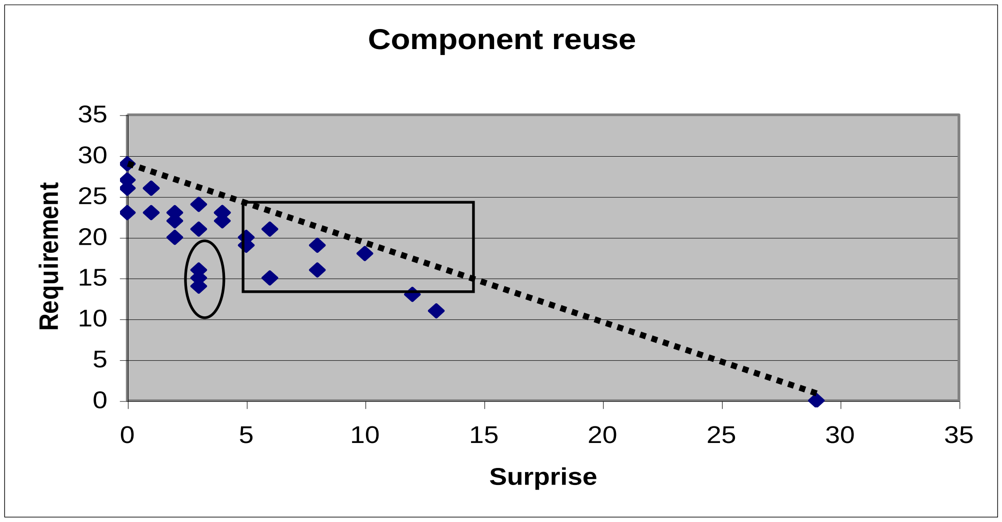


Figure 5 Data from the REBOOT reuse survey in a Lillestøl plot.

The first part - the required factors- are concerned with questions like "Can I find out what the component does?" and "Will the component handle erroneous data?" This is necessary to know in order to reuse the component. However, one large question remains, namely "Can I trust the component" - and this is what the surprise characteristics promise information about. Left to themselves, software developers will only reuse components that they trust - that is why they prefer the ones they have developed themselves. In my opinion, the often cited "Not invented here" syndrome is largely an excuse from component producer who feels that their components have not received the attention and reuse that they feel that they deserve.

Conclusions

Software metrics are important for software development, for software process improvement and for software quality. If we do not consider both the presence ***and*** absence of characteristics, we will miss the information that tells us whether a factor is a surprise, a requirement or just describes something that the customers' consider to be obvious. Most of the work done on SPI and QA has focused on controlling the requirements. The Kano model tells us that this is not enough.

If we want to win new customers and new markets, it is not enough to focus on traditional quality assurance and SPI aiming at removing causes for customer complaints. All this is necessary, but it is not sufficient. In addition we need to improve our understanding of the customers, continuously aiming at surprising and delighting them. This does not necessarily mean new features, although for instance Microsoft has had quite a lot of success with this approach. Other areas where we can surprise our customers are service, user friendliness and interoperability. Each of these factors is an opportunity to grab new markets and outperform our competitors.

The Lillestøl plot will give an easily interpretable picture of the quality characteristics and especially those that are in the transit zones. First and foremost, however, we need to put the customers and their needs in the center of our thinking and learn to understand their needs.

Some developing companies think that the fact that quality is subjective and that its components are in a steady transition from surprise to requirements and then on to the obvious - or to indifferent - is a serious problem. As one manger so aptly put it "I refuse to bow to a definition of quality that is purely subjective. That is not science". Poor guy - he is in for some surprises. The ever-changing concept of quality is an opportunity, not a threat, at least to those who have understood it. It gives us the opportunity to surprise and please the customer and - more important - it gives us the opportunity to tailor a product to a market segment or a group of people instead of clinging to the "one size fits all" idea.

References

[1] Aune, Asbjørn: Quality Oriented Companies (in Norwegian). Ad Notam Gyldendal, Oslo Norway, 1996, ISBN 82-417-0516-6

[2] Karlsson, Even Andre (editor), Software Reuse: A Holistic Approach, John Wiley & Sons, New York USA, 1995, ISBN 0-471-95489-6

[3] Stålhane, Tor: Development of a Model for Reusability Assessment, in Nesi, Paolo (editor), Objective Software Quality, Proceedings from the second symposium on Software Quality Techniques, Firenze, May 29 - 31, 1999, Springer Verlag.

[4] Vonderembse, Mark A. and van Fossen, Tom: Quality Function Deployment, in Madu, Christian N. (editor) Handbook of Total Quality Management, Kluwer Academic Publishers, Dodrecht, The Netherlands 1998, ISBN 0-412-75360-X

[5] Lillestøl, J: Multivariate statistical methods for quality creation: a review, Total Quality Management, vol. 2, pp 291-304

[6] Stålhane, T, Borgersen, P C, and Arnesen, K, In Search of the Customer's Quality View. Proceedings of the AQuIS conference, Firenze, Italy, 1996

[7] Oh, H.L., A Changing Paradigm of Quality, IEEE Transactions on Reliability, vol. 44, no. 2, June, 1995

Appendix

The results from the developer survey of factor importance for reusability are summed up in the table below. The questions that do not fit into the "No, Some, All" scheme or are not significant are left out. In order to grade the quality factors we use the letters s (strong) or w (weak). Thus, for instance R(s) means a strong attitude towards a requirement.

Table 3 Summary of reuse characteristic questionnaires

No	Description	No	Some	All	O, R or S
1	Documentation of the problem that the component solves	--	+	++	R(s)
2	Documentation of the problem that the system solves	--	+	++	R(s)
3	Component header information	-	+	++	R(s)
5	Simple demo examples			+	S(w)
7	Interface description	--	+	++	R(s)
8	Mnemonic constant names	--	+	++	R(s)
9	Code layout standard	--	+		R(s)
10	Input and output sections separation	-		++	R(s)
11	Structure diagrams included	-	+	++	R(s)
14	Code robustness	--	+	++	R(s)
15	List of external dependencies	--	+	+	R(s)
16	Machine dependent code	+	--	--	R(s)
17	Non-standard language constructs	++	--	--	R(s)
18	Standard test cases available		+	+	S(w)
19	Comments in the code	-	+	+	R(w)
20	Mnemonic variable names	-	+	++	R(s)
21	Version and variant history		+	+	S(w)
22	No changes due to corrections		-	-	O(w)
23	Number of critical errors experienced	++	--	--	R(s)
24	Used in many other sites			++	S(s)
25	Development constraints		+	++	S(s)
26	Documentation of the development process			++	S(s)
27	Documented QA procedures used		+	+	S(w)
28	Modern development tools		+	+	S(w)
29	Level of available support	-		+	R(w)
30	QA used during maintenance			+	S(w)
31	No errors after change	+			S(w)

Table 4 Summary of responses for all significant reuse characteristics

No.	Description	R	S	O	I
1	Documentation of the problem that the component solves	23	4		2
2	Documentation of the problem that the system solves	22	4		3
3	Component header information	26	1	1	1
5	Simple demo examples	15	6	1	7
7	Interface description	27		1	1

8	Mnemonic constant names	21	6	1	1
9	Code layout standard	20	2	3	4
10	Input and output sections separation	23	1		5
11	Structure diagrams included	23	4		2
14	Code robustness	26	1		2
15	List of external dependencies	26		2	1
16	No machine dependent code	23		2	4
17	No non-standard language constructs	23	2	1	3
18	Standard test cases available	19	8		2
19	Well commented code	24	3		2
20	Mnemonic variable names	23	4	1	1
21	Version and variant history	16	8	1	4
22	No changes due to error corrections	14	3	6	6
23	No critical errors experienced	16	3	3	7
24	Already used at many other sites	11	13		5
25	Development constraints imposed	18	10		1
26	Documentation of the development process	19	5	1	4
27	Documented QA procedures used	20	5		4
28	Modern development tools used	13	12		4
29	High level of support available	21	3		5
30	QA used during maintenance	22	2	1	4
31	No errors after changes / updates	15	3	1	10

SoPCoM – Model for Evaluation of the Software Processes Complexity

Romana Vajde Horvat, Ivan Rozman, Vesna Lešnik
University of Maribor, Faculty of Electrical Engineering and Computer Science
Institute of Informatics
Smetanova 17, 2000 Maribor, Slovenia
Phone: +386 62 235 5114   Fax: +386 62 235 5134
E-mail: romana.vajde@uni-mb.si; i.rozman@uni-mb.si; vesna.lesnik@uni-mb.si

Abstract *- The method for the estimation of complexity of software process models (SoPCoM) is presented in the paper. The complexity of the software process is estimated on the basis of the complexity of all elements within the model (the artifacts, roles, software tools, equipment, etc.) and the relations between elements and groups of the elements. The method is based on high-level (colored) Petri nets. All elements are represented in the Petri net as colors, tokens, places and transitions. The formalism of Petri nets provides the mechanism for computation of process model complexity. To each element the appropriate attributes are assigned. These attributes are ranged as constant complexity attributes (describing those properties of the process model, which are common for all projects) and represent the complexity of the process model.*

I. INTRODUCTION

It is well known and very often expressed, that "software processes are complex". But when the complexity of the ***software*** process should be measured, very few appropriate models are available - mostly due to lack of utilization of formal methods for software process modeling. [1] Information on software process complexity represents a valuable input for the project planning. The estimation of effort in software development projects is usually based on the characteristics of the product developed within the project. The two most known models - COCOMO [2, 3] and Function points [4, 5, 6, 7]- consider the maturity of the software development process only in general terms.

Generally, the main problem occurring with estimations based on product is the lack of information about the product in early development phases, where estimations are most needed. Supposing the models of development process are defined within the organization at appropriate level of details, the portions of needed effort for different parts of process could be estimated. Knowing the information about effort in early phases could provide reliable estimations for other parts of process. This is the main issue of the SoPCoM (Software Process Complexity Model) presented in following sections.

II. SOFTWARE PROCESS

The term software process[1] unifies all activities which have to be performed in order to develop a software. It includes also the activities for software installation, user training, etc. The methods for implementation of activities depend on the type and content of development projects and technology used. For the same type of projects, the same sequence of activities and the same methods for their implementations are used. The sequence of activities are defined within so called Software process models. For example, the development of OO (object oriented) software can be conducted according to Objectory or Iconix process model. To understand the content of process models some terms should be explained:

1. **Process** - is a group of interrelated steps/activities, leading to common goal, and all of the elements necessary for their execution. Software process, consequently, includes activities for the development of software.
2. **Activity** - is the smallest - atomic action the process. Its structure is not visible to outside. Terms **task** and **process step** are also often used to refer to the atomic actions of the process.
3. **Sub-process - i**s a group of interrelated activities, forming an integral part of the process. They can be treated as independent units (or procedures), usually as pattern of process activities. *Requirements capture* sub-process is a typical example of software processes.
4. **Resources** - are employees and facilities needed to perform specified activity. Types of resources are:

- ***Roles*** - the roles are describing human resources, together with their responsibilities and authorities.
- ***Software* - i**ncludes software applications and tools, supporting or automating any segment of activity. Software is divided into two groups: *active software*(it can activate interaction in process itself) and *passive software*(which is always activated by users).
- ***Hardware* - i**ncludes workstations, servers, printers and other hardware needed within software process.
- ***Infrastructure* -** includes offices, office equipment and logistic, communication and other accessories, needed for software development. Infrastructure is modeled only when special requirements (for example, accessibility, special equipment) are required.

5. **Artifact** - is a product, used/produced within the process. Examples of artifacts: documents created as results of activities, plans, diagrams and other technical products, generated by activities. They are used as inputs for the next activity.
6. **Process model** - is a presentation of specified process. It can be used as a template for executing the real process.

Knowing the basic elements of the process the appropriate method for the representation of their relationship. Different methods are used according to the purpose of the presentation: non-executable paradigms, state-based paradigms, rule-based paradigms, imperative paradigms. See [8] for more details. Among others state-based paradigms the Petri nets are often used in practice.

Basic elements of Petri nets are (see Fig. 1):

transitions – they can be used for presentation of activities performed within the process,

tokens – which can be used to represent the artifacts and resources needed to perform the activity or produced by activity,

[1] ***Software process is a synonym for software development process***.

places – which represent the temporal store for tokens. If a token is placed on a specific place it is evident that previous activity has generated it.

arcs – they represent the path of tokens between places and transitions.

An example of the usage of Petri nets for software process modeling is presented on Figure 1. Activities form the *Test phase* are modeled as transitions (like *Test Plan Preparation*, *Preparation of test cases*, *Preparation of Test Environment*, etc.) For each activity the required inputs and outputs are defined. For activity *Test Plan Preparation* the inputs are *Configuration item* and *Requirements Specification*. These inputs are modeled as tokens (defined according to their type - like type:CI). Each token comes to the transition from a place (P), where it waits until it can be processed. If each token type would come from a separate place, the number of places would be large as well as the number of arcs. Therefore, different types of tokens can be joint at the same place (like at P12). The model, presented in the following chapter, is based on described principles of modeling.

The usage of high-level Petri nets is only briefly described here, more information about it can be found in [9, 10, 11, 12].

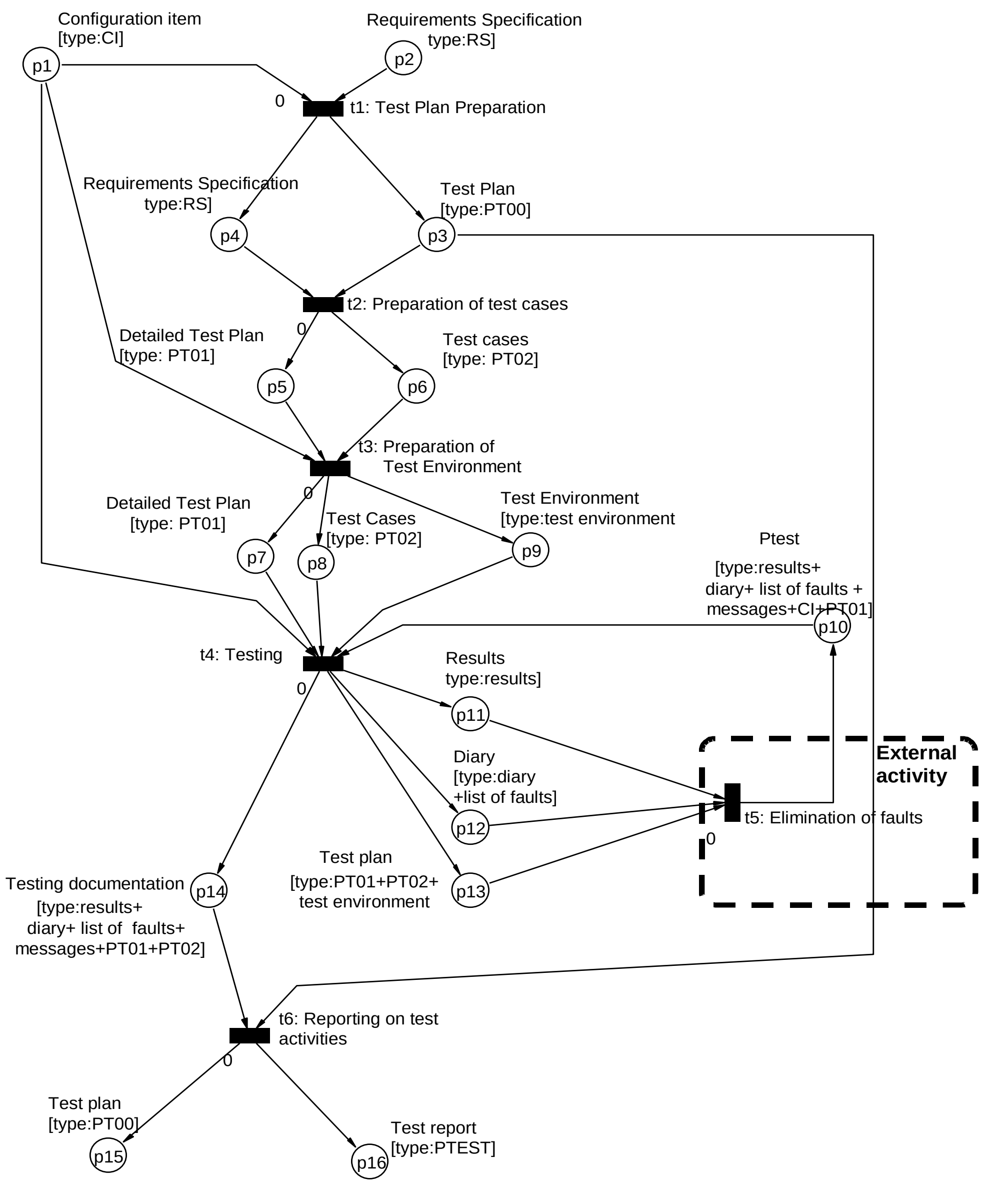


Fig. 1. Test process presented in notation of Petri nets.III. SoPCoM FRAMEWORK

The SoPCoM is based on high-level (colored) Petri nets. It means that each type of artifacts (requirements, glossary, use case model, etc.) and each type of resources (use case modeler, system engineer, etc.) can be represented as a specific color of tokens. For each place the relevant colors of tokens (the relevant artifacts and resources) are defined. The presence of tokens at each place represent that tokens were properly processed within previous activities and are prepared to be used in successive activities. [13]

Each transition (activity) can be performed in different ways – it means that for the transition different combinations of input tokens are needed and different combinations of resulting tokens can be generated. Therefore, the colors of transitions are introduced for each of possible ways to perform the activity.

The arcs between places and transitions (or reverse) are weighted. For each arc the number of tokens of each color transferred form place to transition (or reverse) are defined.

The above described relationship of elements in process model are presented with mathematical formalism within SoPCoM. Still, the mathematical representation of the process model is only the basis on which the SoPCom is founded. This representation provides the mechanism for further evaluations and analysis of the model.

Briefly, the SoPCoM defines the set of attributes which can be used to evaluate the complexity of each artifact and resource and the complexity each type (way of performance) of the specific activity.

The framework of the SoPCoM includes following activities:

1. Definition of artifacts and resources

The list of artifacts and resources is prepared. For the evaluation of the artifacts, roles, infrastructure, SW and HW, different attributes should be used for each of mentioned group. Therefore, the colors of tokens are grouped according to their characteristics.

For the example presented on Fig. 1, the following colors of artifacts are defined:

$$\mathbf{C}_p = [requirements, glossary, vision, UC, UCMo, \sup plementary\_specification, iteration\_plan, UCP, arhitecture\_diagram]$$

The list of groups of artifacts and resources is presented as

$$\mathbf{G}_p = \lfloor g_{p1}, \ldots, g_{pm}, \ldots, g_{pn_{pg}} \rfloor^{\mathrm{T}}.$$

2. Evaluation of complexity of artifacts and resources

Each artifact and resource is evaluated according to appropriate list of attributes and its complexity is saved in vector

$$\mathbf{x}_{c_p s} = \lfloor x_{c_p 1s}, \ldots, x_{c_p ls}, \ldots, x_{c_p n_{c_p} s} \rfloor^{\mathrm{T}}.$$

Table 1 presents the list of attributes for all groups. For each attribute pertaining measurement scale is defined. Table 2 presents the definition of measurement scale for ADEF attribute.

3. Definition of activities

All activities and places are listed in vectors $\mathbf{T}$ and $\mathbf{P}$.

4. Definition of types of activities

For each activity (transition) the set of possible types of its performance is defined. For example, activity *t2:Find actors and use-cases* can be performed in two different ways: in first case it generates supplementary specifications and in second case it does not. In both cases the complexity of activity is different, therefore, the complexity of both cases should be evaluated separately. This is resolved with two different colors assigned to the activity (transition). The probability of execution of each type of activity should be defined. In SoPCoM the colors of transitions are presented in vector $\mathbf{C_t}$ and probabilities are presented in matrix $\mathbf{C}$.

TABLE I. List of groups and pertaining complexity attributes

GROUP	ATTR. ID	DESCRIPTION
1.1.1.1 Artifacts	**ADEF**	Definition
	ARUS	Reusability
	ACON	Configuration management
	AGEN	Generating Artefact Req.
Roles	**REXP**	Experience
	RSPC	Specialization

	RLEA	Leadership
	RCOM	Communication
Software	**SEXP**	Experiences
	SSUP	Support
	SAVL	Availability
	SEXC	Data Exchange
	SALT	Alternative Tools
Hardware	**HAVL**	Availability
	HPER	Performances
	HINT	Hardware: Interoperability
Places	**PAVL**	Availability
	PTMP	Temparature
	PEQP	Equipment
	PLOC	Location

TABLE II. Measurement scale for the ADEF attribute

Attribute: Definition (ADEF)	
0	Not relevant
0.2	Form defined in details – template with detailed instructions and examples exists, no modifications of template needed
0.4	Form defined – template with instructions exists, needed modifications of template
0.6	Basic elements of artifact are defined, no template
0.8	Basic guidelines for generating the artifact are defined, no template
1	Form is not defined, instruction do not exist, template does not exist

5. Evaluation of complexity of each activity type

Similar as for artifacts and resources the complexity of each type of activity is evaluated. The specific attributes are defined for activities. Results are stored as:

$$\mathbf{x}_{c_t s} = \left[x_{c_t 1s}, \ldots, x_{c_t ls}, \ldots, x_{c_t n_{c_t} s} \right]^{\mathrm{T}}$$

6. Definition of input and output constraints for each activity and its types

For each type of activity (presented as color of transition) the input and output constraints should be defined. This means that for each color of transition the number of tokens arriving through each input arc should be defined. The same constraints should be defined for output tokens. Fig. 2 presents the relationship between colors of transition and input/output tokens.

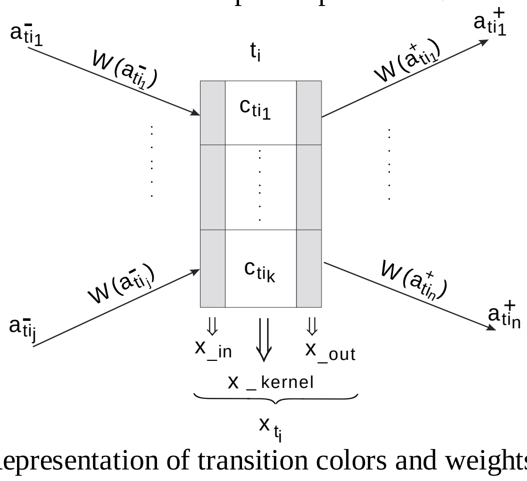


Fig. 2. Representation of transition colors and weights of arcs.

Weights for all arcs are gathered in matrix $\mathbf{Y}^{-}$ and $\mathbf{Y}^{+}$.

7. Definition of artifact and resources influence on complexity

In previous steps the complexity of each separate element was evaluated. Further it is important, how these elements interact within the process. For example, if first activity generates the *use case* and in seven other activities only use this *use case* for input information, the influence of the *use case* on complexity is much more important in first case. To present the influence of the element is presented in the matrix $\mathbf{H}^{-}$ for input tokens and $\mathbf{H}^{+}$ for output tokens.

8. Calculation of complexity of each activity

Considering the relationship between elements, the complexity of each activity is calculates according to following equation:

$$\boxed{\mathbf{x}_{ts}[i] = \sum_{j=1}^{n_{c_t}} \mathbf{C}[i,j] \cdot (x\_in + x\_out) \cdot x\_kernel \cdot 10} \qquad (1)$$

where:

$$x\_in = \sum_{k=1}^{n_p} \mathbf{A}^{-}[i,k] \cdot \sum_{l=1}^{n_{c_p}} \mathbf{Y}^{-}[i,k,j,l] \cdot \mathbf{H}^{-}[\mathbf{D}_p[l]] \cdot \mathbf{x}_{c_p s}[l]$$

is the sum of complexity of input tokens;

$$x\_out = \sum_{k=1}^{n_p} \mathbf{A}^{+}[i,k] \cdot \sum_{l=1}^{n_{c_p}} \mathbf{H}^{+}[i,k,j,l] \cdot \mathbf{x}_{c_p s}[l]$$

is the sum of complexity of output tokens;

$$x\_kernel = \mathbf{x}_{c_t s}[j]$$

is the influence of each color of transition.

9. Calculation of complexity of the complete process model

The complexity of the complete process model is calculated simply as a sum of complexities of separate activities.

III. SoPCoM in practice

The SoPCoM model was used in real software development environment for the *Test process* as shown on Figure 1. The model of the process was first developed in EPC notation by staff of participating organization and then transformed into Petri nets notation by research group. Also needed information on elements of the process model was assured by organization staff. They provided the evaluation of how complex the Test plan (form PT00), Test cases (form PT02), diaries (form diary) and other elements are and how much do they change in each activity. Research group than used these data within the SoPCoM model to compute the relative share of complexity and needed effort for performing each activity within the process model. Results of the SoPCoM algorithm were then compared with real data (amount of hours needed to perform each activity) from four projects. Figure 3 shows the results of comparing the share of needed time to perform testing activities in real situation and results of SoPCoM evaluations.

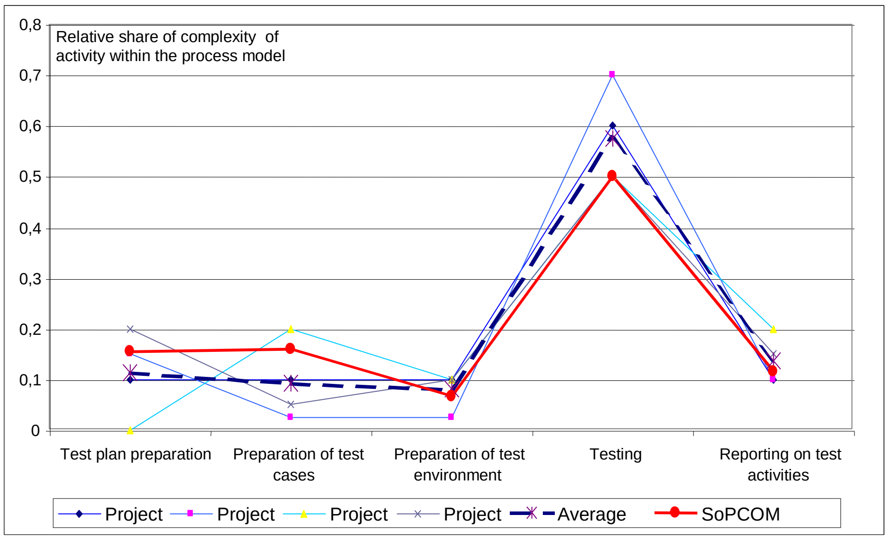


Fig. 3. Comparison of SoPCoM results and real projects effort.

Knowing the process model in details (as shown in steps form 1 to 9) the share of effort for each activity could be evaluated. If the whole software development process would be defined as required in SoPCoM, than for each new project which would use the specified process model, the share of effort needed for each activity would be known. This would improve the project planning process. Further, if activities within the process would belong to specific type (like activities for monitoring of processes) the share of effort needed for each group would be known. The detailed knowledge of the process could be also used as the basis for risk management.

IV. CONCLUSIONS

The most valuable information provided by SoPCoM is the relative share of complexity between separate activities in a process model. The model is at the present time in testing phase in real cases form SW development companies. The main problem with testing the model is that processes in real organizations are rarely defined in details, needed for the evaluation by the model. The software organizations, which implement the software process improvement activities, are the potential users of the SoPCoM model because their processes are mature. Further, the appropriate SW support is needed to perform the SoPCoM evaluations easy and efficiently enough. The SoPCoMTool is being developed for that purpose at our institute.

[1] R. Park, W. Goethert, J. Webb, Software Cost and Schedule Estimating: A Process Improvement Initiative, CMU/SEI-94-SR-03, Software Engineering Institute, Pittsburg, PA.

[2] Barry Boehm et al. , Cost Models for Future Software Life Cycle Processes: *COCOMO 2.0*, Anals of Software Engineering Special Volume on Software Process and Product Measurement, Science Publisher, Amsterdam, Nederlands, 1995.

[3] COCOMO II Model Definition Manual, University of Southern California, Center for Software Engineering, 1998, http://sunset.usc.edu/research/COCOMOII/index.html

[4] David Garmus, David Herron, Measuring the software process (A practical guide to functional measurements), Yourdon press, Upper Saddle River, New Jersey, 1996.

[5] C. Behrens, Measuring the Productivity of Computer Systems Development Activities with Function Points, IEEE Transactions on Software Engineering, November 1983.

[6] IFPUG, *Function Point Counting Practices: Manual Release 4.0*, International Function Point Users' Group, Westerville, Ohio, 1994.

[7] Albrecht, A.J., Measuring Application Development Productivity, Proceedings of Joint SHARE, GUIDE and IBM Application Development Symposium, October 1979, stran 83 – 92

[8] Alfonso FUGGETTA, Alexander WOLF, *Software Process*, Trends in Software 4, John Wiley & Sons, Chichester, 1996.

[9] Jean-Marie Proth, Xiaolan Xie, Petri Nets - A tool for Design and Management of Manufacturing Systems, John Wiley¸Sons, Inc. New York, 1996.

[10] High-level Petri Nets - Concepts, Definitions and Graphical Notation, Committee Draft ISO/IEC 15909, October 2, 1997, Version 3.4

[11] Wolfgang Reisig, Grzeorg Rozenberg, Lectures on Petri Nets I: Basic Models, Springer, Berlin, 1998.

[12] http://www.dsi.unimi/Users/Tesi/trompede/petri/nets/TPN.html

[13] Vajde Horvat Romana, Software Process Complexity, PhD Thesis, University of Maribor, Maribor, March 2000.

PROBE: Development of a European Benchmark on IT Acquisition Processes

Yingxu Wang
Dept. of Electrical and Computer Engineering
University of Calgary
2500 University Drive, Calgary, Alberta, Canada T2N 1N4

IVF Centre for Software Engineering

Argongatan 30, S-431 53, Molndal, Gothenburg, Sweden

Yingxu.Wang@acm.org

Abstract: *PROBE is a European SPRITE S2 research project for developing a European benchmark on IT acquisition processes. The objective of the PROBE project is to establish European-wide benchmarks and improvement experience repository of acquisition practices, which enable organizations to compare and better manage their acquisition improvement activities. PROBE and the European IT acquisition process benchmarks also enable a new approach to benchmark-based process assessment in the important IT acquisition area.*

In this paper, the PROBE IT acquisition process model and assessment method are described. Based on these, the structure of European IT acquisition process benchmarks and query functions are presented. PROBE derives a set of IT acquisition process benchmarks at European, national, sector, and organizational levels. By using the European acquisition process benchmarks, comparative studies and performance analysis for an organization can be carried out, and potential strengths and improvement opportunities can be identified. Industry case studies and assessment results are reported with regard to the European benchmarks.

Key words: Software engineering, software process, IT acquisition process, benchmarks, benchmark-based process improvement

1. Introduction

PROBE is a European research project for benchmarking and improvement experience repository for European IT acquisition practices [1-6]. The goal of the PROBE project is to establish European-wide benchmarks and improvement experience repository of acquisition practices, which enable organizations to compare and better manage their acquisition improvement activities.

PROBE utilizes a standard methodology to assess the acquisition processes and practices of over 100 organizations, and creates a benchmark database that allow organizations to compare their own acquisition processes to regional and industry sector averages. Software organizations will thereby be able to better prioritize, motivate, and focus their acquisition improvement actions based on formulae, which have already proved successful in other organizations, as well as to better prepared and motivated to identify areas of competitive advantage.

This paper describes the PROBE IT acquisition process model and process assessment method. The structure of the European IT acquisition process benchmarks and the design of the query functions are presented. Derived European benchmarks are reported and status of European IT acquisition process improvement are analyzed.

2. The PROBE IT acquisition process model

A set of over 100 PROBE assessment results provides the basis for IT acquisition process benchmarking in Europe. This section introduces the PROBE IT acquisition process model, including the PROBE process model, capability model, and capability determination methodologies.

2.1 The PROBE Process Model

The PROBE process model is developed with 4 process categories, 9 practice areas, and 66 key activities as shown in Table 1 [1].

Table 1. The PROBE assessment model

ID.	Category	Key practice areas	Key activities
ACQ	Acquisition		
ACQ.1		Acquisition needs	1.1- Acquisition policy, 1.2- Acquisition strategy, 1.3-Benefits analysis
ACQ.2		Acquisition preparation	2.1-Acquisition planning, 2.2-Risk identification, 2.3-Potential solutions
ACQ.3		Requirements definition	3.1-Technical requirements, 3.2-Non-technical requirements, 3.3-Contract requirements, 3.4-Financial requirements, 3.5-Project requirements
ACQ.4		Contract award	4.1-Invitation to tender, 4.2-Supplier qualification, 4.3-Tender evaluation, 4.4-Contract establishment
ACQ.5		Contract performance	5.1-Supplier monitoring, 5.2-Acquisition documentation, 5.3-Acquisition testing, 5.4-Life-cycle costs analysis, 5.5-Contract changes, 5.6-Contract closure, 5.7-Cost re-evaluation, 5.8-Operational use
ACQ.6		Acquisition environment	6.1-Suppler relationships, 6.2-User relationships, 6.3-Financial management, 6.4-Open system infrastructure, 6.5-Maintenance policy, 6.6-Post delivery support, 6.7-Training
SUP	Support		1-Documentation, 2-Quality assurance, 3-Verification, 4-Problem tracking
MAN	Management		1-Project management, 2-Configuration management, 3-Risk management, 4-Process management, 5-Human resource management
ME	Project		

T	metrics		
MET.1		Infrastructure goals	1.1-Team co-operation, 1.2-Team involvement, 1.3-User satisfaction, 1.4-User participation, 1.5-User complaints, 1.6-Number of suppliers, 1.7-Supplier responsiveness, 1.8-Project information
MET.2		Financial goals	2.1-Discount levels, 2.2-Project extension, 2.3-Contract terms and conditions, 2.4-Return on investment, 2.5-Additional funding, 2.6-Unexpected purchases, 2.7-Project overrun
MET.3		Process goals	3.1-System performance, 3.2-Effort, 3.3-Training

2.2 The PROBE Process Capability Model and Process Attributes

A set of 9 attributes has been identified in the PROBE capability model [2] for evaluating each practice activity described in the PROBE assessment model as shown in Table 2. The attributes focus on important aspects of practice performance and the level of practice capability, supplemented by a number of evaluation aids.

Table 2. The PROBE capability model

ID.	Attribute	Focus	Questions
AT1	Performed	Is the task performed?	• Is the scope of work defined? • Are identifiable work products associated with a task produced?
AT21	Planned and tracked	Is the task planned and tracked?	• Are the objectives of the task identified? • Are key activities and milestones of the task defined? • Are resources and responsibility for performing the task assigned? • Is progress of the task monitored according to a defined plan?
AT22	Product integrity	Are the documents produced by the task appropriately managed, configured, and under change control?	• Are requirements for the documents (e.g. specifications, plans, code) defined? • Are dependencies among the documents identified? • Are documents appropriately identified? • Are configuration and changes to the documents defined and controlled? • Are the documents verified and adjusted to meet the defined requirements?
AT31	Documented	Is the task defined and documented?	• Is a description and requirement for the task documented? • Is the task documented and under configuration and change control? • Is appropriate guidance for execution of the task defined? • Is the task performed in accordance with its definition?
AT32	Quality achieved	Are suitable validation, verification, review and auditing activities implemented?	• Are roles and responsibilities of quality activities for the task assigned? • Are quality activities carried out according to defined activities within the documented task and for any applied quality system? • Are records maintained to demonstrate quality achievement?
AT41	Usage	Is the usage of the task proven, accepted and stable?	• Does the defined task have proven coverage and tailorability? • Does the defined task have proven performance and capability? • Is the defined task accepted by those who are impacted by it? • Is the defined task stable?
A	Skills	Is the task performed	• Are requisite knowledge, skills and competence

T42		by staff with appropriate skills, competence and training?	identified for responsible staff? • Is a strategy established towards developing competent staff for the task? • Is the scope and aim of training defined and planned?
A T51	Measured	Is the performance and capability of the task quantitatively measured?	• Is efficiency of the task monitored and evaluated? • Is effectiveness of the task quantitatively analysed? • Are the results of reviews, audits and evaluations analysed to identify trends and root cause of problems?
A T52	Improved	Does the task have an optimum environment and optimum operational satisfaction?	• Is the environment in which the task is performed regularly reviewed for potential improvements? • Is the defined task refined with experience? • Are best practices and new technologies regularly evaluated with a view to incorporating into the defined task? • Are potential improvement actions identified, prioritised, planned and implemented?

For enabling the rating of each process with quantitative measures, a PROBE practice attribute rating scale is defined as shown in Table 3, where *F, L, P,* and *N* stand *for fully, largely, partially,* and *not* adequate respectively.

Table 3. PROBE attribute rating scale

Symbol	Description	Rating
F	Fully achieved	86% to 100%
L	Largely achieved	51% to 85%
P	Partially achieved	16% to 50%
N	Not achieved	0% to 15%

2.3 PROBE Algorithm for Practice Attribute Scoring

A PROBE tool is developed to automatically calculate process capability levels by using an algorithm as described below for transforming the practice ratings in the form of F/L/P/N into a percentage value.

Assuming the weights, W_i, i = 1 … 9, for the 9 attributes are:

W1 (AT1) = 1
W2 (AT21) = 2
W3 (AT22) = 2
W4 (AT31) = 3
W5 (AT32) = 3
W6 (AT41) = 4
W7 (AT42) = 4
W8 (AT51) = 5
W9 (AT52) = 5
(1)

and the values for an attribute rating, R_j, j = 1 …5, are:

R1 = F = 0.93
R2 = L = 0.67
R3 = P = 0.33
R4 = N = 0
R5 = N/A = F = 0.93
(2)

The score of a practice k, S_k, for all attributes is defined as a sum of the weighted rates, i.e.,

$$S_k = f * [\sum_{i=1\ldots9} (W_i * R_j)] = 3.7 * [\sum_{i=1\ldots9} (W_i * R_j)] \quad (3)$$

Where, *f* is a factor that unifies the scores in a range of 0 … 100.

For example, if a practice is rated as *F* for all attributes, the score of this practice is:

$$S_1 = f * [\sum_{i=1\ldots9} (W_i * R_j)] = 3.7 * 29 * 0.93 = 100$$

If a question is rated as *N* for all attributes, the score is:

$$S_2 = f * [\sum_{i=1\ldots9} (W_i * R_j)] = 3.7 * 29 * 0 = 0$$

Therefore, the final score of a PROBE process will be located between 0 and 100.

3. Design of PROBE benchmarks on IT acquisition processes

A European database of acquisition process benchmarks is developed by the PROBE project. This section describes the PROBE IT acquisition benchmarking database requirements and query definitions. A benchmark database schema, a set of derived benchmarks, and a number of query functions are defined and illustrated.

3.1 PROBE Benchmark Database Schema

The PROBE benchmark database is described by a main record (schema), a number of fields, and a set of query functions [5].

The main record structure of PROBE benchmark database, *PROBEBenchmark*, consists of demographic information, acquisition questions, and project metrics questions as defined below:

PROBEBenchmark ::= **RecNo** | DemographicQuestions | AcqQuestions | ProjectMetricsQuestions;
(4)

Where, the items separated by '|' are the *fields*, and the fields in bold are the *keys* for sorting and index in query.

A primary key for the benchmarking database is defined as *RecNo*. The range of the numeric key is between 1 to 9999.

RecNo ::= 1 … 9999 (5)

There are 9 generic questions that cover key demographic information such as assessee's country, sector, organizational size, acquisition department size, type of acquisition, ISO 9001 certification and years, acquisition costs, and annual spend on acquisitions.

DemographicQuestions ::= **RefNo** | **Nation** | **Sector** | **OrgSize** | **AcqSize** |
TypeOfAcq |ISO9K | YearOfISO9K | AcqCosts |
AnnualSpend | Remarks (6)

38 practices have been identified as described in Table 1, covering the practices of acquisition, support and management. The structure of these practices is defined below:

AcqQuestions ::= **AcqQNo** | KeyWord | Question | Explanation | Attribute1 |
Attribute2 | … | Attribute9 | **Score**;
AcqQNo ::= 1 … 39
(7)

Two keys have been identified as *AcqQNo* and *Score*. Algorithms for deriving the score for each question based on the ratings of its attributes are defined in Formula 3.

18 IT acquisition project metrics questions have been designed as shown in Table 1, covering the goals of infrastructure, financial and process. The structure of this questions is as follows:

ProjectMetricsQuestions ::= **ProQNo** | KeyWord | Question | Answer;
ProQNo ::= 1…18
(8)

3.2 Types of European Benchmarks

A benchmark of process is defined as an average reference value that the process statistically performs in a given sector or a given region. A number of PROBE benchmarks on European IT acquisition processes are derived as described below.

European benchmarks: This is a benchmark at European level for the average scores of all countries, sectors, size-groups, and categories of questions.

National benchmarks: These are national benchmarks derived from the whole European benchmark. 20 countries' benchmarks have been identified, supplemented by a benchmark of the rest outside Europe.

Sector benchmarks: 22 industrial sectors have been identified for establishing the European sector's benchmarks.

Size benchmarks: 6 sizes of companies as shown below have been classified for establishing the European size-based benchmarks.

Combined benchmarks: In addition to the above benchmarks, combined benchmarks of nation-sector (462), nation-size (126), sector-size (132), and nation-sector-size (2772) can be produced.

3.3 PROBE Benchmarking Query Support

PROBE benchmarking database is designed to support the following queries:

- Current benchmarks
- An organisation vs. a benchmark
    - An organisation's position in a size-group, sector, nation, or Europe
- An organisation's potential strengths with regard to the benchmarks
- An organisation's potential improvement opportunities with regard to the benchmarks

The following subsections describe these queries and their functions.

3.3.1 Current benchmark report

Up-to-date benchmarks can be reported as defined below:

- the European benchmark
- upto 21 national benchmarks
- upto 22 European sector benchmarks
- upto 6 European size-group benchmarks
- upto 462 nation/sector benchmarks
- upto 126 nation/size benchmarks
- upto 132 sector/size benchmarks
- upto 2772 nation/sector/size benchmarks.

3.3.2 Comparative analysis report

For an organization *X*, the following comparative analysis reports can be produced on request:

- X vs. the European benchmark
- X vs. a national benchmark
- X vs. a sector benchmark
- X vs. a size-group benchmark

3.3.3 Position analysis report

For an organization *X,* the following position analysis report can be produced on request:

- X's absolute (number x) and relative (top of percentage) positions in the European benchmark
- X's absolute/relative positions in a national benchmark
- X's absolute/relative positions in a sector benchmark
- X's absolute/relative positions in a size group benchmark

3.3.4 Potential strengths analysis report

For an organization *X*, the following strength analysis report can be produced on request:

- Practice statuses that are above the European average
- Practice statuses that are above a national average
- Practice statuses that are above a sector's average
- Practice statuses that are above a size-group's average

3.3.5 Potential improvement opportunity analysis report

For an organization *X*, the following improvement opportunity analysis report can be produced based on weakness analyses on request:

- Practice statuses that are below the European average
- Practice statuses that are below a national average
- Practice statuses that are below a sector's average
- Practice statuses that are below a size-group's average

A web-based PROBE benchmarking tool is developed based on specifications of this section. The PROBE benchmarking tool provides benchmarking services through the Internet for registered users. In the initial phase, 105 PROBE assessment results generated pan Europe have formed the basis of the PROBE benchmarking database.

4. Derived European benchmark on IT acquisition *processes*

This section reports and analyzes major European benchmarks derived by the PROBE project. The sector-oriented European benchmark and the status of process improvement in Europe are analyzed based on the data documented in [5, 6].

4.1 European Benchmark of IT Acquisition Processes

The benchmark of performance of IT acquisition processes in Europe is shown in Figure 1. This figure provides a high level summary of the status of acquisition practices performed in Europe.

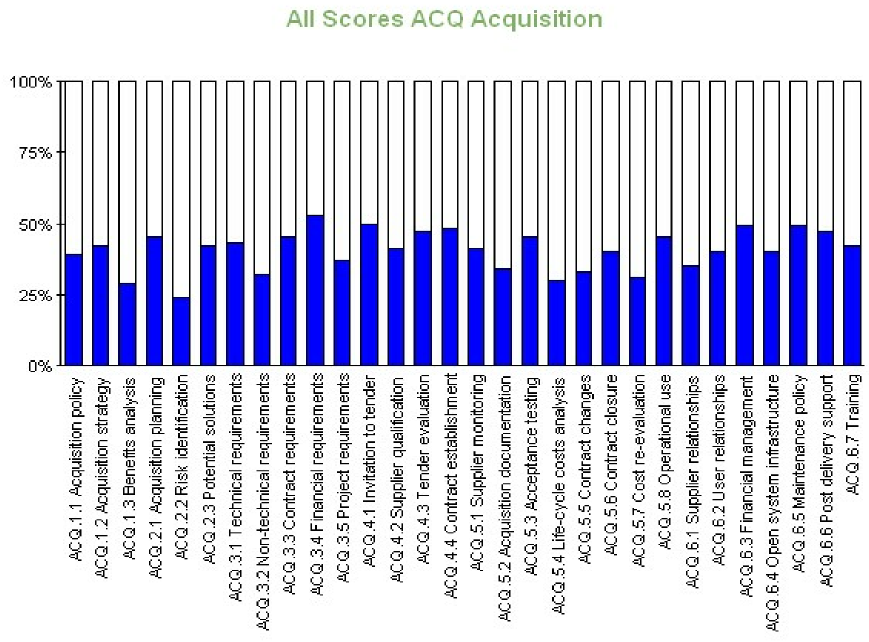


Figure 1. European benchmark of acquisition process performance

In Figure 1 the six acquisition process categories as defined in Table 1 have been benchmarked. In average, the performance of the processes is at 35% level. This shows that the industry performance is far from reaching the perfect level. The acquisition processes with the best performance are ACQ3.4 – Financial requirements, ACQ4.1 – Invitation to tender, ACQ6.3 – Financial management, and ACQ6.5 – Maintenance policy. The acquisition processes that are less performed are ACQ2.2 – Risk identification, ACQ1.3 – Benefits analysis, and ACQ5.4 – Life cycle cost analysis.

4.2 Status of IT Acquisition Process Improvement in Europe

This subsection shows the benchmarks of acquisition process improvement activities in European organizations. Two attributes, measured and improvement, are focused on to analyze the activities in acquisition processes improvement.

The distribution of levels of acquisition process formality is summarized in Figure 2. Figure 2 shows that: a) 31% of the acquisition processes in Europe are "fully" or "largely" measured; and b) 69% of the acquisition processes in Europe are "not" or "partially" measured. For improvement of the acquisition processes, 37% of them are "fully" or "largely" addressed; and 63% of them are "not" or "partially" committed.

Generally, only 1/3 of the European organizations has adopted process measurement and improvement in acquisition and procurement. The majority rest has not been improvement-oriented based on process measurement and analysis. This fact indicates that quantitative process measurement and analysis toward continuous process improvement should be significant emphasized in the European industries.

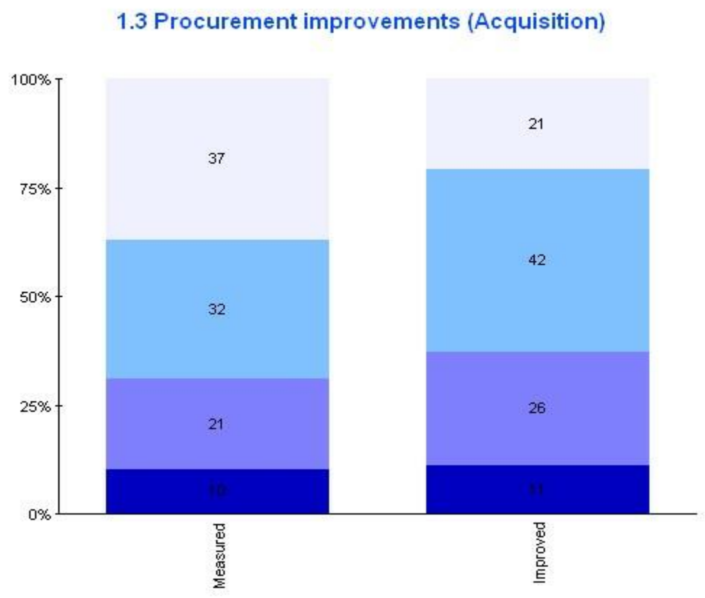


Figure 2. Distribute of procurement improvements

5. Conclusions

The PROBE project has identified the needs for developing a European benchmark on IT acquisition processes. For enabling organizations to compare and better manage their acquisition improvement activities, the aim of the PROBE project has been designed to establish European-wide benchmarks and improvement experience repository of acquisition practices.

By using the European benchmark of IT acquisition processes developed by PROBE, organizations can benefit: (a) for maximum return on investment from acquisition improvement activities; (b) for increased confidence in and motivation for acquisition improvements; and (c) for reuse proven solutions developed in other organizations, to ease identification and lower risk associated with improvements in their own acquisition processes.

One of the key values of PROBE benchmarks is the information it provides to users concerning acquisition practices in other European regions. Organizations can determine where they need to improve to be more competitive at a European level. Industry sector information allows organizations across Europe to compare their practices against companies in the same sector in other European regions.

Acknowledgements

This project is partially funded by European Commission through the SPRITE S2 Programme Project No. 98/502056. The author would like to acknowledge the support of European Commission, all partner organizations of the project, and the user organizations that participated in the establishment of the European IT acquisition benchmarks.

References

[1] Wang, Y., Dorling, A., Pitette, G., and Hansen, S. (1999), D.2.2.1: PROBE IT Acquisition Assessment Questionnaire Model, *Technical Report of PROBE*, European Commission SPRITE S2 Research Project 99/502056, pp.1-21.

[2] Wang, Y. and Dorling, A. (1999), D.2.2.2: PROBE Attribute Rating Model, *Technical Report of PROBE,* European Commission SPRITE S2 Research Project 99/502056, pp.1-7.

[3] Wang, Y. and Dorling, A. (1999), D.2.2.3: PROBE Assessment Report Template, *Technical Report of PROBE,* European Commission SPRITE S2 Research Project 99/502056, pp.1-12.

[4] Wang, Y. and Dorling, A. (1999), D.2.2.4: PROBE Self-Assessment Tool, *Technical Report of PROBE,* European Commission SPRITE S2 Research Project 99/502056, pp. 1-47.

[5] Wang, Y. and Dorling, A. (1999), D.3.1: PROBE Benchmarking Database Requirements, *Technical Report of PROBE,* European Commission SPRITE S2 Research Project 99/502056, pp.1-11.

[6] Wang, Y. (2000), D.3.2: PROBE European Benchmarking Services, *Technical Report of PROBE,* European Commission SPRITE S2 Research Project 99/502056, pp.1-10.

[7] Wang, Y. and Hansen, S. (2000), D.8.1: PROBE European Benchmark Report, *Technical Report of PROBE,* European Commission SPRITE S2 Research Project 99/502056, pp.1-201.

About the Authors

Yingxu Wang is Professor of Software Engineering in Dept. of Electrical and Computer Engineering at The University of Calgary, and project manager with the Center for Software Engineering at IVF, Gothenburg, Sweden. He received a PhD in software engineering from the Nottingham Trent University / Southampton Institute, UK. He is a member of IEEE, ACM, and ISO/IEC JTC1/SC7, and is Chairman of the Computer Chapter of the IEEE Swedish Section. He was a visiting professor at Oxford University. He is the lead author of a recent book on *Software Engineering Processes: Principles and Applications*, and he has published over 100 papers.

CBS

Copenhagen Business School

Organised by

ISCN

STTP

DERA

Arbeitskreis Software-Qualität Franken

ASQF

SINTEF Telecom and Informatics

THE SWEDISH INSTITUTE OF PRODUCTION ENGINEERING RESEARCH

IVF INSTITUTET FÖR VERKSTADSTEKNISK FORSKNING

DELTA

ASQF (www.asgf.de) · **DELTA** (www.delta.dk)
DERA (www.dera.gov.uk) · **ISCN** (www.iscn.ie) · **IVF** (www.ivf.se)
SINTEF (www.sintef.no) · **STTF** (www.sttf.fi)

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