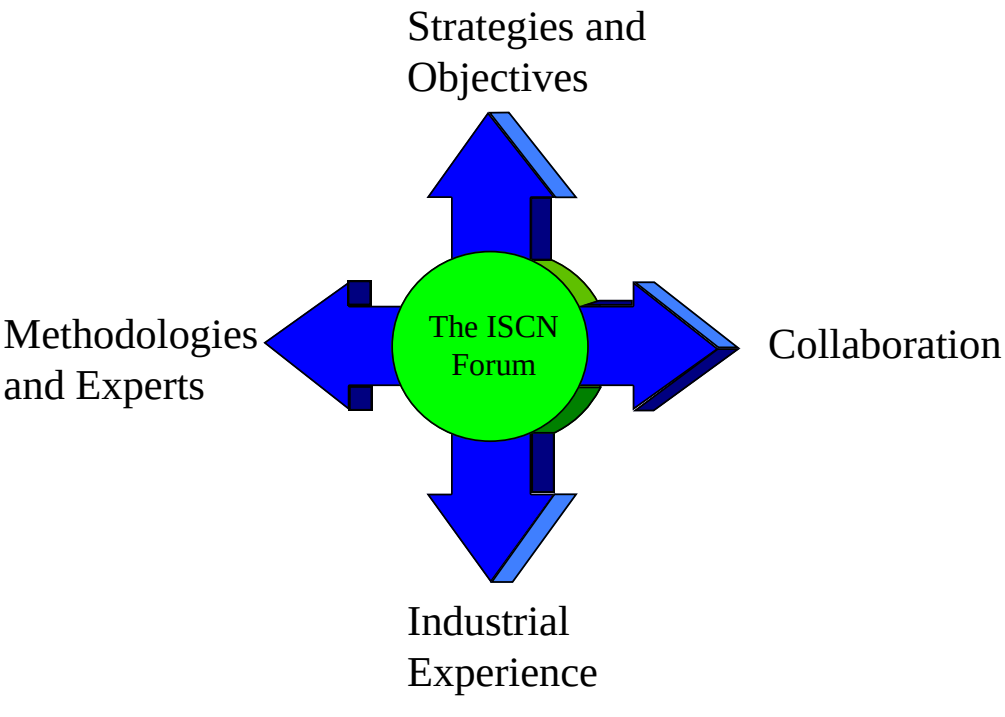
ISCN 96

Metropole Hotel
Brighton, 3-5 December 1996

ISCN´96 Practical Improvement of Software Processes and Products

1996 Theme - A Multimethod View of Process Improvement



INTRODUCTION

Strategies, Co-operations, and Networks

SP´96 Plenary Session: Assessment and Goal Driven Improvement

Experience With Measurement Approaches

Experience With Process Assessment and Improvement Models

Experience With CMM

Process Improvement Combined Approaches

Software Development Processes

Preface

ISCN (International Software Collaborative Network) was formed in 1994. Its initial mission was twofold: to organise annual conferences on software process improvement, and to create a consortium of software process experts capable of providing a wide range of software process consultancy services.
Successful conferences were held in 1994 (ISCN94, Dublin) and 1995 (ISCN95, Vienna). These brought together three types of players: large users (companies with major software process improvement programmes), methods providers (small companies offering specific methods and techniques in process/product analysis, measurement, and improvement), and individual experts (independent consultants, experts from companies acting on behalf of their companies in ISCN projects, or academic researchers).
ISCN´96 is the third conference. All ISCN conferences focus on practical experience in improving software processes and products. They are not academic in nature, but concentrate on tried and tested methods which can be presented in quantitative terms. The main goal is to create a discussion culture which allows participants to discuss „How will this work in my own organisation ?“.

We invite you to join the process improvement discussions and presentations at ISCN´96 and to help ISCN establishing a culture in which organisations work together, exchange know how, and share effort, knowledge and risk to work on process improvement problems.

Dr Micheal Mac an Airchinnigh
President, ISCN Ltd.

Dr Richard Messnarz, Programme Chair
Director, ISCN ltd.

ISCN´96 Programme Chair

Richard Messnarz, Director, ISCN Ltd.

ISCN´96 Programme Committee

Micheal Mac an Airchinnigh (IE)
Gualtiero Bazzana (It)
Miklos Biro (HU)
Christophe Debou (B)
Khaled El Emam (D)
Eamonn Mac Guinness (IE)
Wilhelm Schäfer (D)
Jean Martin Simon (F)
Tilo Messer (D)
Colin Tully (UK)

ISCN´96/ SP´96 Organising Committee

Chair: Colin Tully (SPI´96)
Alec Dorling (SPICE´96)
Richard Messnarz (ISCN´96)
Wilhelm Schäfer (ICSP4)
Caroline Summer/John Herriot (Meetings Management)

ISCN´96 Organising Committee

Chair: Richard Messnarz
Reinhard Urban
Petra Watzke

ISCN´96 Session Chairs

Strategies, Co-operations, and Networks
Micheal Mac an Airchinnigh, President ISCN (Ireland)

SP´96 Plenary Session: Assessment and Goal Driven Improvement
Colin Tully, CTA (UK) and ISCN

Experience With Measurement Approaches
Miklos Biro, President John von Neumann Society, Sztaki (Hungary)

Experience With Process Assessment and Improvement Models
Wilhelm Schäfer, University of Paderborn (Germany)

Experience With CMM
Claude Laporte, President Montreal SPIN, Oerlikon Aerospace (Canada)

Process Improvement Combined Approaches
Bernhard Posch, APS (Austria)

Software Development Processes
Mechthild Rohen, European Commission (Belgium)

Short Summary of ISCN´96 Sessions

Strategies, Co-operations, and Networks

Collaborative approaches and potentials for co-operation with the ISCN network are discussed. The president of the Montreal SPIN presents information about strategic process engineering activities in Quebec in Canada. A manager from 3SE presents and discusses business potentials with Indian companies from Bangalore. And the president of the John von Neumann society presents the Hungarian quality scene and offers the establishment of further co-operations with Hungarian industry. The session focuses on new business potentials and how to exploit them by using new principles such as collaborative networks, virtual companies, and learning organisations.

SP´96 Plenary: Assessment and Goal Driven Improvement

This session shows the experience of two major industrial companies in implementing SPI initiatives: First Siemens's SPI corporate approach is introduced, highlighting how they are using a large scale assessment methodology which combines the strengths of CMM and BOOTSTRAP. in addition to that Siemens runs a large improvement programme OPAL which transfers the assessment results into priority driven improvement projects. Then the application of a goal-oriented, measurement driven approach to software process improvement (ami) will be illustrated through two case studies one coming from Alcatel and a second one provided by Objectif Technologie based on experiences with defence companies. The two insist on the importance of linking the initiative to business issues. This session will include a panel discussion organised by Colin Tully who is the ISCN management board member responsible for establishing a SPIN of users in future.

Experience With Measurement Approaches

This session presents various methods for quantitatively analysing software processes and how to use these measures for identifying best practices, potentials for improvements, and bottlenecks. An experience study from Brameur discusses a basic set of metrics which can be used for effective process management. Nowadays there are hundreds of metrics and hundreds of researchers identifying further metrics, and Brameur will focus on a minimum set of metrics to be used. A study from the Fraunhofer IESE Institute will deal with the GQM paradigm and how to practically employ it for establishing an experience base. And Onion, an Italian company offering advanced IT solutions, presents the results of an ESSI PIE which established and measured guidelines for process improvement in Internet based information management.

Experience With CMM

This session brings together a group of experts who have long lasting experience with CMM. Ken Dymond has published the book "A Guide to the CMM" and is well known as an SEI lead assessor having worked in many different countries and continents. Eamonn Mac Guinness of AIMware has experience with many CMM based improvements (some more successful than others). Using the SEI CMM AIMware were assessed to ISO 9001 / TickIT within 5 months of start-up. Eamonn presents the IDEALsm method for achieving maturity

increases with the SEI CMMsm. (IDEAL and CMM are service benchmarks of CMU). Francois Coallier is the leading expert in Trillium and is the Vice President of Bell Canada, and he will illustrate how Trillium is used as a CMM based benchmark for the Telecom sector. Denis Roy is the director of ASEC (Applied Software Engineering Center) in Montreal which is a representative of SEI in Canada and which organises a Canadian wide improvement programme.

Experience with Process Assessment and Improvement Models

Khaled El Emam was a lead researcher at CRIM in Montreal (he is now at Fraunhofer IESE in Germany) who is largely involved in the SPICE trials evaluating and analysing the results from the trials. He is also the editor of the Software Process Newsletter. He and Denis R. Goldenson from SEI present the results from the SPICE phase 1 trials. Jean Martin Simon is a senior consultant at CISI, France, and is largely involved in the SPICE project. He has significant experience with process models based on the ISO 12207 process modelling standard which also formed the basis for the SPICE process model. He gives practical tips about how to use SPICE. Pasi Kuvaja is the director of the BOOTSTRAP institute and Richard Messnarz was a lead researcher in the ESPRIT project BOOTTSTRAP and closely co-operates with Pasi Kuvaja as a BOOTSTRAP lead assessor in industry. They present the new version of BOOTSTRAP which is SPICE compliant and will discuss practical experience with BOOTSTRAP.

Process Improvement Combined Approaches

Recently a consortium of ISCN members and partners have started a new initiative funded under the EU Leonardo da Vinci programme. PICO (Process Improvement Combined apprOach) develops a comprehensive set of tutorials that cover process improvement from analysis to success. PICO is a modularised product which does not rely on distinct methods and tools. PICO seminars are based on learning by doing and will focus on re-usable experience. PICO shall start a learning process in organisations affecting different target groups from top managers to engineers. This will lead to the establishment of a self learning culture and organisation.

Software Development Processes

Elisabeth Kauba from Siemens presents the results of an ESSI process improvement experiment analysing the potentials of Re-Use and how to employ re-use strategies. Roman Cunis from MAZ Hamburg presents the results of an ESSI process improvement experiment which introduced object oriented methodologies and techniques into the organisation. Kurt Walk (retired, manager of the IBM Vienna SW Lab) discusses about how to apply object oriented principles not only to software but to organisational work processes and scenarios. And Andrew Butterfield discusses about practical experience with the use of formal approaches to achieve reliable software and the role of formal methods in software process improvement.

ISCN - The International Software Collaborative Network

Richard Messnarz, Director Process Development, ISCN Ltd., Ireland
Micheal Mac an Airchinnigh, Trinity College, Dublin, Ireland
Colin Tully, CTA, UK
Miklos Biro, Sztaki, Budapest, Hungary
I.S.C.N. Ltd.
Florence House, 1 Florence Villas
Bray, Co. Wicklow, Ireland
Tel. +353 1 286 1583, Fax. +353 1 2865078
email: iscn@genrix.ie, URL: http://www.iol.ie/~iscn/info

Introduction

ISCN (International Software Collaborative Network) was formed in 1994. Its initial mission was twofold: to organise annual conferences on software process improvement, and to create a consortium of software process experts capable of providing a wide range of software process consultancy services [2].

Successful conferences were held in 1994 (ISCN94, Dublin) and 1995 (ISCN95, Vienna). These brought together three types of players: large users (companies with major software process improvement programmes), methods providers (small companies offering specific methods and techniques in process/product analysis, measurement, and improvement), and individual experts (independent consultants, experts from companies acting on behalf of their companies in ISCN projects, or academic researchers).

In 1995 ISCN established a business firm functioning as a co-ordination office to secure funding and to provide an organisational and management infrastructure for innovative improvement projects on behalf of its members and partners [12]. ISCN is currently organising the 1996 conference (ISCN 96, Brighton), preparing a major book for publication next year, and developing training material.

By the end of 1996, ISCN plans to extend its mission and mode of working. The intention is to build a multi-firm, multi-nation, multi-industry and multi-method learning organisation, in which the partners are able to form collaborative groupings for problem solving and training purposes on an as-needed basis. The benefits sought are synergy, win-win, reduced risk, rapid skill development, and a culture in which partners are able to improve faster by working together than they could on their own.

ISCN headquarters will act as both entrepreneur and facilitator. As entrepreneur it will play a leading role in establishing, and securing funding for project consortia, among partners, to solve problems arising in process improvement initiatives. Projects will, for example, develop and field-test new methods or approaches, improve or integrate existing ones, or develop and field-test training material for the acquisition of new skills.

Project funding will come from a mixture of internal sources (shared costs among partners), and external sources (such as EU funding programmes). As facilitator, ISCN HQ will provide infrastructure support for projects, in the form of (for instance) common processes and tools for management, communication, reporting, meetings, and control.

The ISCN service portfolio already contains more than twenty different methods offered by about forty partners. A number of projects have already been already launched or are at the proposal stage.

This article describes ISCN´s mission, goals, strategies, services and projects, and deals with the structure of a technology transfer bridge between service and technology providers and large users.

Mission, Strategy and Goals

The ISCN banner is variety and diversity. The mission of ISCN is to satisfy the needs of its partner firms for highly qualified expert support of their software process assessment and improvement initiatives. There are several methods for software process assessment, measurement, and improvement. ISCN encourages the combined use of such approaches and methodologies by using effective teamwork and collaboration based on win-win situations for all, the customers, the experts, and ISCN.

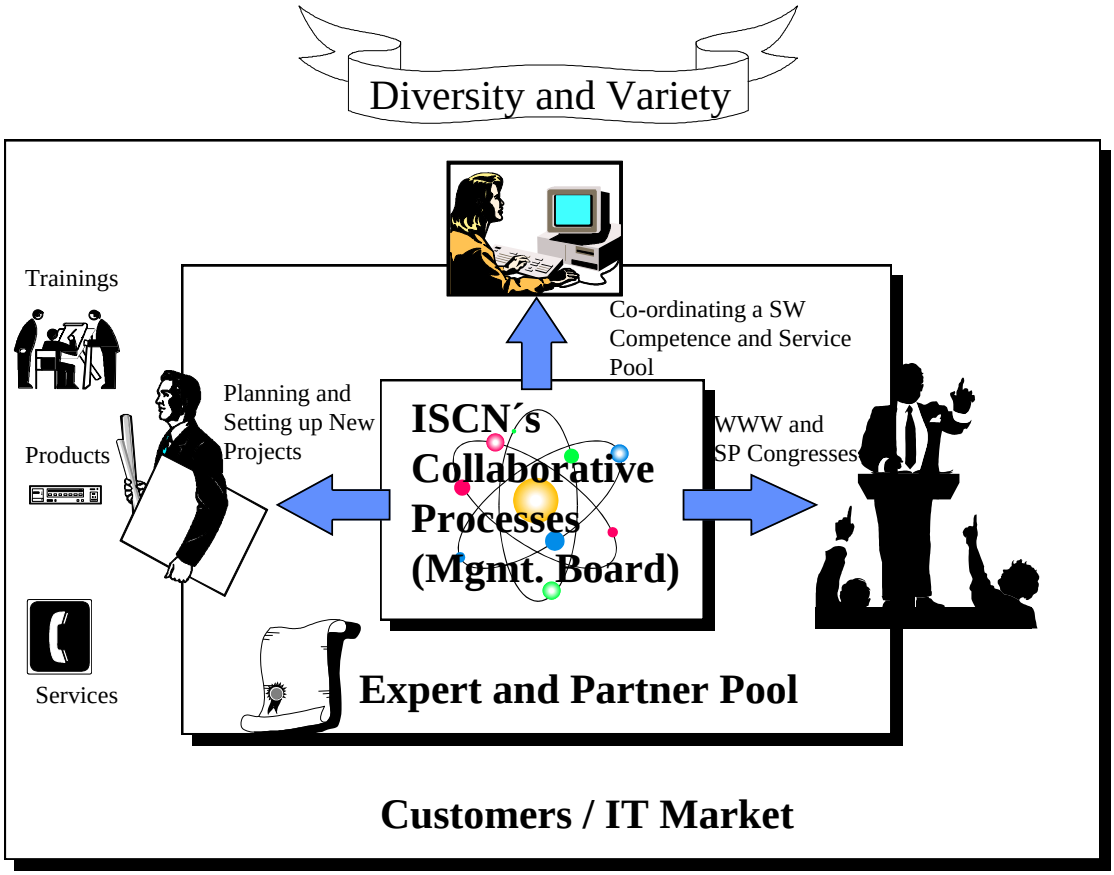


Fig. 1: The ISCN Architecture

A key asset of ISCN is its pool of experts who represent a wide range of approaches and methodologies allowing a synergetic combination of the skills most suitable to the specific requirements of the customer. ISCN has been and will be committed to the highest professional traditions by applying quality assurance and continuous improvement to its own consulting processes, while it is also ready to re-engineer these processes if there is an opportunity for better satisfying the needs of business partners.
In this spirit, ISCN exploits the capabilities offered by most recent information and communication technologies to enable the most efficient organisation of its own activities, to bring about radical changes in the ways customers are served. These technologies coupled by its strong commitment allow ISCN to become a virtual part of its business partner´s

organisation. ISCN is by consequence an extended enterprise which stretches the traditional boundaries of professional consulting.

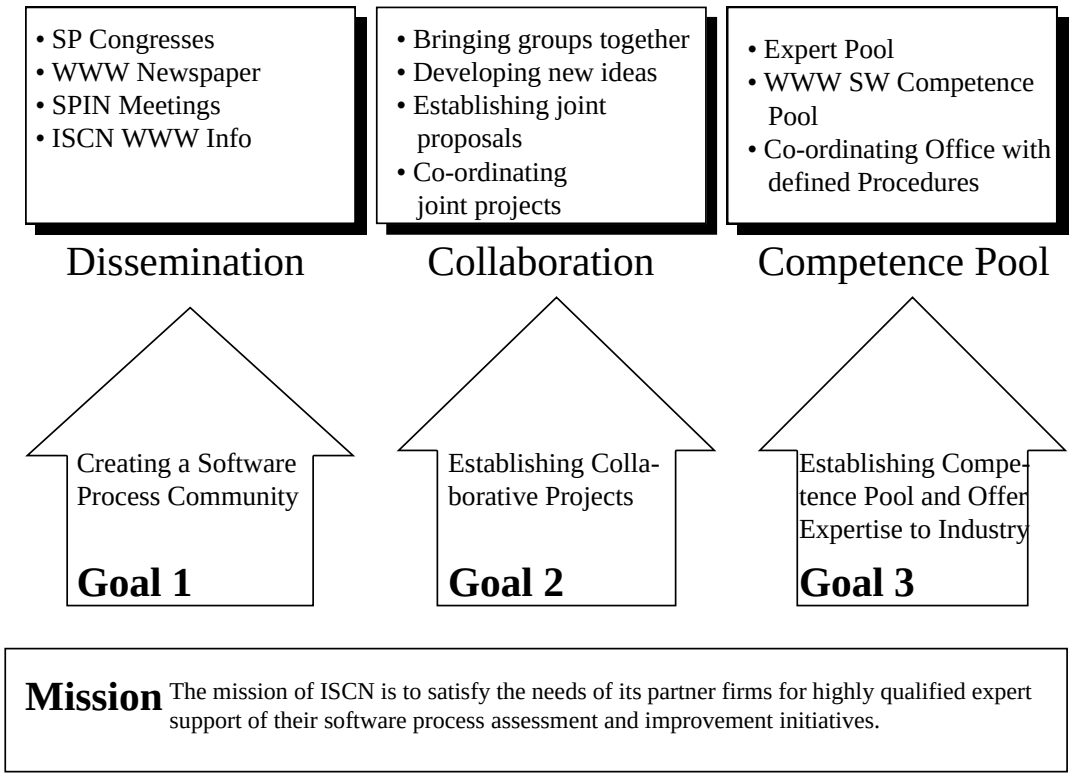


Fig. 2 : ISCN´s Collaborative Processes

ISCN pursues three major business targets (Fig. 2): Dissemination, Collaboration, and the establishment of a Competence Pool. For each of these goals ISCN has been developing products and services for infrastructure support from 1994 up to now (see section 3).

Activities supporting the goal of "Creating a Software Process Community" are

- ISCN conferences about "Practical Improvement of Software Processes and Products" as part of an annual Software Process Congress in co-operation with other major process improvement conferences
- a process improvement newspaper on WWW which containing three top articles from each conference in which ISCN is involved

Activities supporting the goal of "Establishing Collaborative Projects" are

- using the annual conference as a point for identifying new ideas and forming groups of partners with shared interests
- installing email and discussion groups and supporting the process of formulating new ideas in project proposals
- Co-ordinating and supporting the process of planning, estimating, and controlling the projects.
- Providing the partners with facilities for up-to-date communication and quality assurance procedures

Activities supporting the goal of "Establishing a Competence Pool of Experts" are

- designing and distributing an ISCN leaflet which contains a service portfolio
- using defined procedures (ISCN process model) for expert selection, team establishment, and project control.
- establishing collaborative agreements with SPI experienced companies based on win-win situations
- establishing a WWW pool of SPI experienced organisations

ISCN´s Infrastructure Support Processes

To ensure that the goals are achieved and that the activities related to the goals are efficiently carried out a number of internal development projects have been performed between 1994 and 1996. These projects aimed at the development of process models [11] and software to automate, support, and standardise best practise work processes for the required activities outlined in section 2.

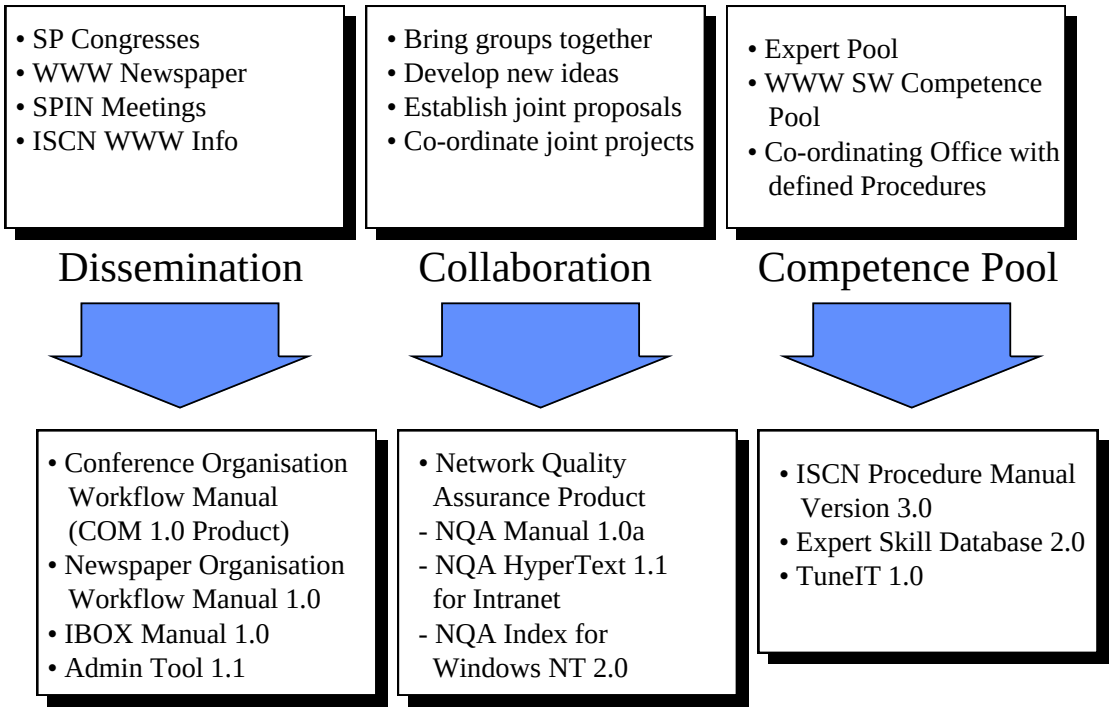


Fig. 3: Products Supporting the ISCN Collaborative Processes (see below projects)

The ***ISCN Procedure Manual Version 3.0*** describes procedures such as integration of new members, team selection, project establishment and co-ordination, and expert pool maintenance.

The ***Expert Skill Database Version 2.0*** was developed with Access 2.0 using the ESA PSS 05 Software Engineering Standards and automates parts of the ISCN procedures. Each expert´s details are stored in the Expert Skill Database. After the customer has described his problems/requirements the expert skill database supports the mapping of these customer problem/requirements data onto expert skill data. The database also supports queries based on restrictive data for engineers, consultants, trainers, and managers such as experience,

publication range, cost, languages, etc. This way the database acts like an expert system virtually representing the ISCN office.

The ISCN ***Conference Organisation Workflow Manual Version 1.0 (COM)*** describes a business and marketing driven approach for organising conferences. This approach is different from organising academic conferences because it mainly focuses on principles such as aggressive marketing, selecting top people and establishing an industry driven workshop-style event, and professionally designing and planning (including cost estimation) events. The manual was used for ESI-ISCN´95 in Vienna and is currently being employed for organising ISCN´96-SP´96 in London/Brighton.

This Network Quality Assurance (NQA) project resulted in three products: 1. The ***NQA manual Version 1.0a***, 2. The ***NQA HyperText Version 1.1***, 3. The ***NQA Winword Macro Version 2.0*** with a set of templates.
The NQA manual differentiates between software and quality system development procedures. Four types of documents are used: planning documents, product related documents, quality related documents, and maintenance related documents. For the work and document flows guidelines and procedures are described.
The NQA HyperText brings the key aspects of the manual into an Intra- or Internet and provides a computer supported index so that employees can inform themselves about required standards and procedures. e.g. if testing is to be performed, what procedures are to be followed, and which standards are to be kept, and which documents are to be produced, and which metrics must be collected.
The Winword Macro is to be installed on a Windows NT server so that users of Winword are provided with an NQA menu from which they can select proper templates for documents. The templates also contain pre-defined bookmarks for important terms

ADMIN version 1.1 is an X-based tool that runs on Linux/Unix supporting the following functions: building up collections for HTML pages, modification of this structure, automatic generation of structure files, provision of an open tool database which can be easily extended, including an overview of the established structure.

The current version of the ISCN home page at http://www.iol.ie/~iscn/info has three clusters of information: The ISCN Network, the ISCN Newsletter, and a pool of experienced SPI companies who are members or project partners of ISCN.
The ***IBOX (Internet information BOX) manual version 1.0*** describes ISCN´s server architecture, the business potentials and the networking approach, guidelines for how to establish and use clients to access information, a documentation of the ADMIN 1.1 tool, and HTML design guidelines. Currently a version 2.0 is being produced. It will form the basis for a training that introduces ISCN members and partners to up-to-date communication and management environments.

The ***Newspaper Organisation Workflow Manual Version 1.0*** describes a process model to market, design, and edit a WWW newspaper. The procedures of NOM and the ADMIN 1.1 tool are currently being used to design a WWW newspaper for the SP 96 congress (section 4.1).

Acknowledgements

We gratefully acknowledge the support of the EU Comett programme for partly funding the work on the ISCN Modelling, Expert Skill Database, Conference Organisation Manual, and the Network Quality Assurance Procedures, and the support of the EU Leonardo programme for partly funding the work on the Internet Information Box and the WWW Newspaper.

Typical ISCN Projects

As outlined in the previous sections there are typical collaborative projects and win-win situations for three business targets: dissemination, collaborative projects sharing cost and effort for product and service development in a consortium of members and partners, and one-stop shopping consulting for large industry.

Dissemination

ISCN has recently set up a collaborative agreement with a group of process improvement workshops and conferences for organising one large annual SP (Software Process) congress in Europe. This strategic agreement resulted in the SP´96 congress in December 1996 in London / Brighton in which 4 conferences are co-operating.

- ISCN´96 - Practical Improvement of Software Processes and Products
- SPI´96 - Software Process Improvement
- ICSP 4 - International Conference on Software Processes
- SPICE ´96 (Software Process Improvement and Capability dEtermination)

This congress approach will be continued over the next few years. In the organisation we follow a distributed and modular approach. Every conference establishes its leaflets, programmes, and proceedings separately. The congress co-ordinating board then integrates the leaflets and programmes to set up one congress programme with parallel conferences that are interfaced by plenary sessions. All partners share the dissemination, marketing, and on-site costs, and distribute the profit by equal shares. Thus the risk is reduced, the costs for each partner are cut down, and a congress approach seems to be more attractive.

Collaboration

PICO (Process Improvement Combined apprOach) Project

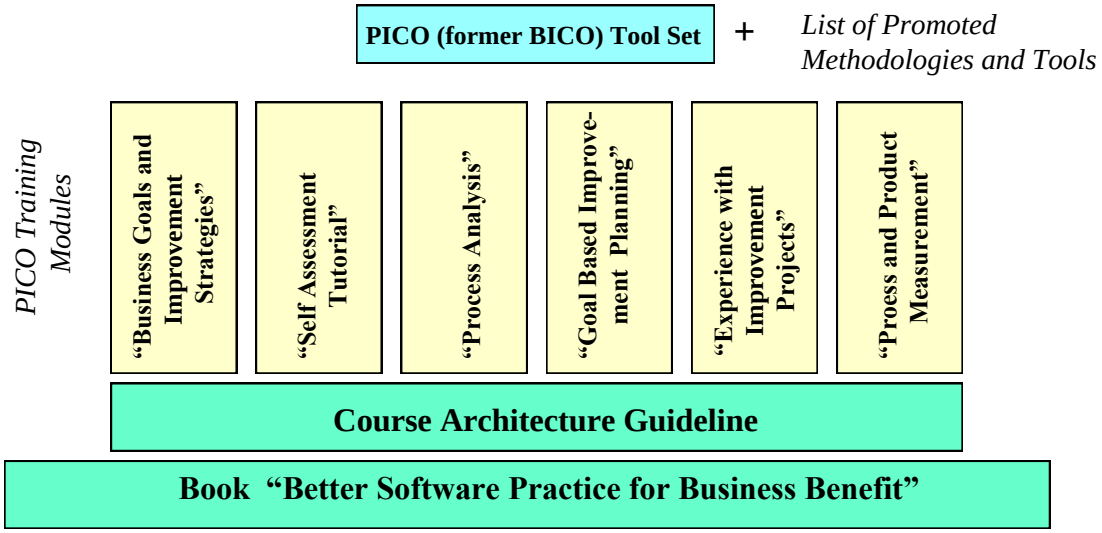


Fig. 4: The Product Architecture of the PICO Project

APAC (Austrian Product Assurance Company) who has experience with the management of aerospace projects is the project manager. ISCN acts as a technical co-ordinator supporting the project communication and the work package design. Project partners are the AIMware, ami user group, APS Austria, Brameur, CISI, Hibernia, Leansoft Oy, and Q-Set Ltd. Additional partners contributing to a major book about process improvement are Alcatel, CRIM Montreal, Colin Tully Ass., Festo, Fraunhofer IESE, Italtel, K&M Technologies, Onion, Siemens, and Sztaki.

The PICO [16] project is developing a comprehensive set of tutorials that cover process improvement from analysis to success. PICO is going to be a modularised product which does not rely on distinct methods and tools. PICO seminars will be based on learning by doing and will focus on re-usable experience. PICO will also promote the tools developed in BICO "Benchmarking & ISO Combined". Together with a group of leading European companies the PICO partners are writing a major book "Better Software Practise for Business Benefit: Principles and Experience" which will be used as a reference material and information pool for the training.

PICO will support the self learning processes of large companies (Fig. 5). Fig. 4 shows that PICO is producing a complete service package (training modules, book, analysis tool, and consulting base) which can either be used to organise open workshops for SMEs or to train training managers of large companies who implement a learning process in their own organisations.

TRAINING MODULES	TARGET GROUPS	Language
• Business Goals and Improvement Strategies Workshop	Division Head, Company Head, IT Executives	Short and Siginificant Business Language
• Process Analysis Workshop • Goal Based Improvement Planning Workshop	Software Engineering Process Groups, Consultants, Improvement Teams	Manager´s Language
• Self Assessment Tutorial • Workshop About Experience With the Implementation of Improvements • Process and Product Measurement Workshop	Project Managers, Quality Managers, Practitioners	Practitioner´s Language

Fig. 5: The Learning Process Addressing Different Target Groups

The project started in December 1995, and it is planned to have a first presentation of the approach at the SP´96 congress in London/Brighton in December 1996, to publish the book in spring 97, and to carry out field tests of a first version of the training modules in 1997.

PASS (Pay Roll Accounting and Settlement System) Project

The PASS project represents an ESSI process improvement experiment [14] carried out by Memolux, a leading Hungarian Budapest-based software company. ISCN is acting as a European partner responsible for transferring the project experience and results back to the EU and supporting the project in the establishment of a measurement programme and the dissemination of results.
The project partners are ISCN supporting the dissemination and exploitation in Europe, Memolux as the software company performing the PIE, and Sztaki (the Computer and Automation Institute of the Hungarian Academy of Sciences) as the software engineering expert partner of Memolux. The project starts in autumn 1996, and will have a duration of about 18 months.

The Component Software and Outsourcing Model

ISCN is currently establishing a strategic co-operation with Eastern European organisations for delivering high qulaity software at very competitive prices for customers in Europe, Canada, and the US.
ISCN is acting as a co-ordinator evaluating the capability of Eastern European firms, setting up price lists and estimation procedures as a basis for the establishment of outsourcing contracts, and co-ordinating the projects. Those Eastern European firms who show sufficient capabilities will receive modul contracts and develop software following quality procedures defined e.g. in the NQA products.
Customers can contact ISCN who then selects a number of Eastern European firms that can deliver the product with the required quality within the defined budget frame. This model will lead to products and services which can be developed and offered at very competitive prices.
A new ISCN director´s position has been estalished for working on this market potential. First procedures and prices will be presented by beginning of 1997.

Consulting

At the time being rather large companies are interested in ordering sets of services (service packages) from ISCN. In a big German company, for instance, ISCN conducted a Bootstrap assessment analysing an SPU with 3 projects. Within three months the assessment was performed, a maturity profile was calculated and evaluated, and an action plan comprising a set of 6 improvement projects was installed [13].
The implementation of one of the improvement projects required further training in the fields of process modelling, as well as analysis and establishment of effective work scenarios. In the assessment, for example, we found that re-use is a key aspect in this organisation but up-to-then it had been based on personal contact rather than a defined work scenario. So the organisation ordered a process modelling training to learn how to identify activities, workflows, roles, results, and resources, how to conduct interviews, and how to design a model which is applicable to the organisation and establishes an effective re-use process.
During the implementation of the improvement projects the organisation realised that the objectives of the improvement projects must be made measurable to be able to evaluate if the goals have been achieved. Here it is of key importance that the goals are consistent with the company´s business goals. This is why the GQM approach (using the ami approach) was employed to establish a metricated goal tree in which the projects´ goals were the leaves of the tree and the company´s business goals represented the root [1].
We are still continuing co-operation with this customer and are approaching SPICE [7] compliance and combining the assessments with the goal driven ami approach [5].

At present the ISCN service portfolio comprises over 20 different improvement methodologies and any combination is available in a service package. (see http://www.iol.ie/~iscn/info under ISCN/Services).

ISCN Membership

ISCN membership is generally available on a personal basis. However, this PERSONAL MEMBERSHIP usually leads to the involvement of the expert´s organisation into ISCN´s activities. ISCN also establishes CORPORATE MEMBERSHIPS with organisations based on win-win agreements:

- An organisation may offer its services through ISCN´s service portfolio and if successful customer contacts are established via ISCN, ISCN earns a provision based on a marketing agreement.
- An organisation may provide ISCN with a license to offer a service. ISCN uses certified experts to perform the service for customers. Then either ISCN pays a license fee or the license provider earns parts of ISCN´s turnover related to the services sold.
- An organisation may establish a collaborative agreement with ISCN to participate in projects and initiatives of ISCN members and partners who share effort and cost to develop products and services.

This means that ISCN emphasises a business driven approach in which it earns profit if its corporate members profit via ISCN. The above corporate membership requires that at least one representative of the organisation - a key contact to ISCN - has gone through the certification procedure for ISCN experts.

Examples of win-win agreements are:

If ISCN does a Bootstrap assessment together with Leansoft Oy, a full member of the Bootstrap Institute and a partner of ISCN, ISCN does the marketing, does the assessment either together with Leansoft (sharing the profit) or using certified assessors form the experts skill database (paying a certain percentage back to Leansoft).
Companies offer training (e.g. TickIT, SEI, etc.) through ISCN´s service portfolio. If the marketing is successful ISCN establishes the contact and the business and earns 10 to 15% provision.
ISCN forms groups of partners who perform projects together and ISCN does part of the expert work and supports project co-ordination and dissemination.

Currently a new type of membership for large users is being discussed. This membership might include

- access to the products and services on which the ISCN infrastructure processes (section 3) are based
- a tailored and adapted service package for large users based on the service pool of well experienced SPI organisations and experts
- an annual workshop of large companies in which SPI experiences are exchanged and possible joint strategies are discussed.

Status of Member- and Partnership

At present individuals or companies and organisations are involved with ISCN in three ways: as members, partners, or supporters. Members are experts who have gone through the ISCN expert certification procedure (providing references from projects, training, etc. for certain expert fields) and represent their organisations in different ISCN projects, initiatives, and conferences. Partners are organisations who perform projects with ISCN (Leonardo, ESPRIT, ESSI, collaborative agreements) but whose experts have not gone through the ISCN certification procedure so far. Supporters are organisations who mainly support ISCN in its activities (e.g. partly funding the dissemination, partly funding projects, field testing results of projects, etc.) but are not partners in a project consortium with ISCN.

The ISCN Management Board consists of:

- Micheal Mac an Airchinnigh, Trinity College, IRL
- Colin Tully, CTA, UK
- Miklos Biro, IT Consult, HU
- Richard Messnarz, ISCN

Certified members are:

- Gualtiero Bazzana, Onion, Italy
- Mikos Biro, IT Consult, Hungary
- Christophe Debou, ami User Group, Belgium
- Ken Dymond, Process Inc., US
- Eamon Mac Guiness, Aimware, IRL
- Richard Messnarz, ISCN, IRL
- Fran O´Hara, Q-SET, IRL
- Hans Scherzer, APAC, A
- Jean Martin Simon, CISI, F
- Cynthia Wise, Process Inc., US

Project Partners are:

- APS Austria, Bernhard Posch, A
- Brameur, Eric Trodd, UK
- IVF and SPICE, Alec Dorling, S
- Q-SET, Anne Downey, IRL
- Hibernia, John Stewart, IRL
- K&M Technologies, Andrew Butterfield, IRL
- Leansoft Oy, Pasi Kuvaja, Fin
- Meetings Management, John Herriot, UK
- Memolux, Janos Ivanyos, HU
- QUEST, Susanne Lanzerstorfer, A
- Sztaki, Miklos Biro, HU
- University of Paderborn, Wilhelm Schäfer, D

Supporters are:

- FESTO, Gerhard Rutschek, A
- FUEVA, Juan Vincente Garcia Manjon, E
- Objectif, Annie Combelles, F
- Politecnico di Torino, Maurizio Morisio, Italy
- Siemens, Axel Völker, Thomas Mehner, Tilo Messer, D
- etc. (more than 15 additional ones)

Here the partners of 3 additional collaborative projects are not listed. The projects start with beginning of 1997, so the group will increase very soon.

Fig. 6: The ISCN Service Portfolio, 1996

The Business Driven Sustainable Growth Model

The current situation in IT industry shows that most management failures are due to inefficient capacity planning [8] [10], and in many cases the cost are underestimated, the market is more competitive than expected, and the identification and establishment of new business which should bring return on investment (ROI) becomes a game theory problem [17].
ISCN pursues an approach which could be described as "business driven sustainable growth model". ISCN started off with a small team from the business demands in the software process sector. The first demand we identified was to establish a discussion platform for industry to exchange know-how in the fields of process analysis, process modelling, process improvement planning, and implementation of improvement. This led to the ISCN conference series about "Practical Improvement of Software Processes and Products", and it proved sufficient that one key player under the ISCN name took the role of the conference organiser and programme chair. From the conferences we identified an industry demand for accessing a pool of process improvement experts and implementing combinations of different improvement approaches which led to the establishment of a service portfolio and an expert pool, and it was sufficient to have a co-ordinating manager who in addition to the conference organiser worked under the ISCN name. From the combined consulting projects new project ideas emerged and we identified the need for forming groups of partners who share the effort for product and service development. Due to the fact that this is a large collaborative business potential ISCN was established as a business firm with managers co-ordinating the expert pool and projects etc.

This means that ISCN did not invest more than the available budget and business demand, but by identifying more and more industry demands and business potentials the co-ordinating firm has grown step by step [3]. And this leads to a bottom-up definition of the institute´s structure driven by the customer and business demands.

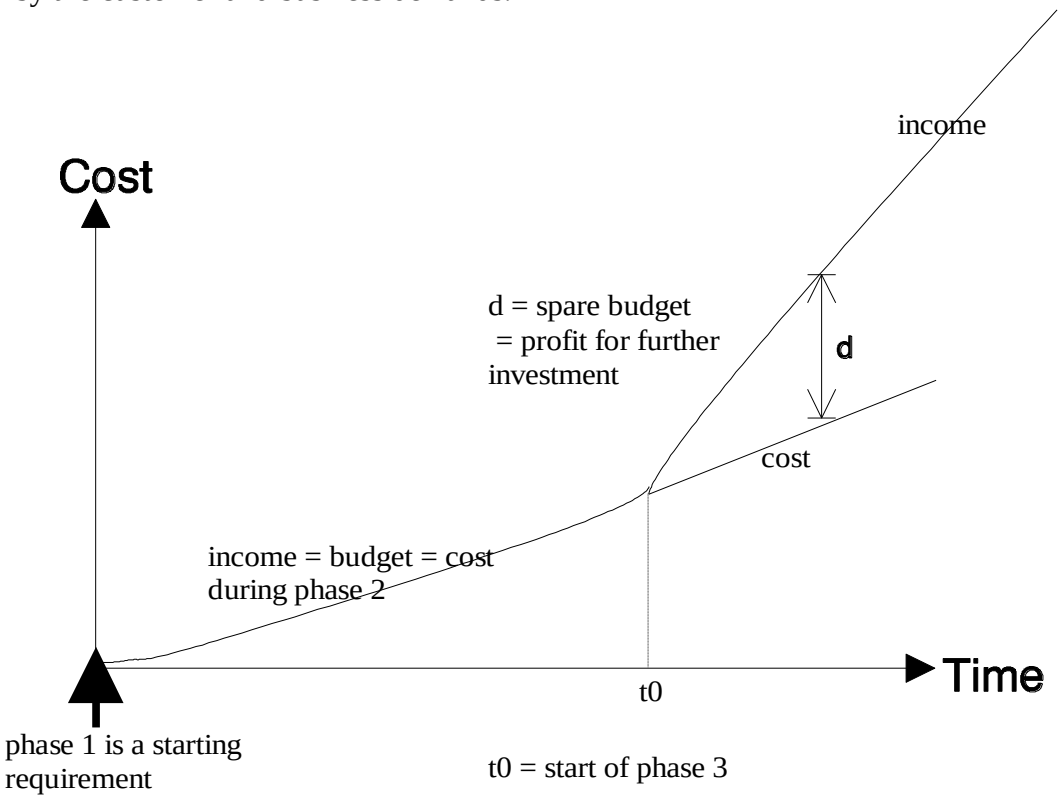


Fig. 7: The Sustainable Growth Model

A business firm following the sustainable growth model runs through three major phases:

- Phase 1 = kick Off phase: The entire process can only be started if a first business demand is identified and can be exploited.
- Phase 2 = growing phase: With the exploitation of one business demand the income is used to identify the next business demand and exploit it. Due to the fact that each exploitation leads to a new business line of the firm all the income is invested in further employment. This way the organisation grows for some time with zero profit, but it is continuously growing with nearly zero risk.
- Phase 3 = profit phase: As soon as the different business lines have led to a stable institute architecture with reliable business partners the growth parameter is stabilised and the income is grows faster than the employment rate which leads to profit.

Future Goals

ISCN´s future goals are to work on collaborative concepts, to identify and exploit further business potentials, and to adhere to the model of "sustainable growth". All activities should be directly influenced and driven by customer, market, and member demands.
As outlined above the collaborative bridge between large users, methodology providers and experts will be a key challenge in the future and we believe that such a multi-nation, multi-method, and multi-industry based approach will fit the European market very well.

References

[1] Basili V., An Experience Factory, in: *Proceedings of the International Conference on Lean Software Development*, Stuttgart, Germany, Oct. 1992

[2] Biro M., Kugler H-J., Messnarz R., et. al., BOOTSTRAP and ISCN - A Current Look at a European Software Quality Network, in: *The Challenge of Networking, Proceedings of the CON´93 Conference*, Oldenbourg, 1993

[3] Boehm B.W., A Spiral Model of Software Development and Enhancement, in: (eds.) Nahouraii E., Butler J.T. Kavi K., Petry F.E., Richter C., Tham K., *Software Engineering Project Management*, pp. 128-142, Comput. Soc. Press of the IEEE, Washington, DC, USA 1988

[4] Chroust G., Computer Integrated Work Management, in (ed.) Mittermeir R., *Shifting Paradigms in Software Engineering*, pp. 4 -13, Springer Verlag, Wien, New York, Sept. 1992

[5] Debou C., ami - A New Paradigm for Software Process Improvement, in: *Proceedings of the ISCN´94 Conference on Practical Improvement of Software Processes and Products*, ISCN Ltd., Dublin, Ireland, 1994

[6] Debou C., Fuchs N., Haux M., ami - A Taylorable Framework for Software Process Improvement, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd., Dublin, Ireland, 1995

[7] Dorling A., SPICE - Software Process Improvement and Capability dEtermination, Software Quality Journal, Volume 2, 1993, pp. 209-224.

[8] Genuchten M., Why is Software Late ? - An Empirical Study for Delay in Software Development, *IEEE Trans. Software Engineering*, pp. 582 - 590, June 1991

[9] German Interior Ministry, German V-Model, Bonn, August 1992

[10] Glass R.L., The Software Crisis - Is it a Matter of Guts-Management, in: (ed.) Reifer D.J., *Software Management*, Fourth Edition, IEEE Computer Society Press, Los Alamitos, 1993

[11] Kellner M. I., A Method for Designing, Defining, and Evolving Software Processes, in: *Proceedings of the European Software Engineering Process Group Conference 1996*, Software Engineering Institute, Carnegie Mellon University, May 1996

[12] Messnarz R., Mac an Airchinnigh M., Decrinis P., Felmann R., Penney D., ISCN - Managing Expert Networks, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[13] 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

[14] Rohen M., ESSI - The European Systems and Software Initiative, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[15] Rutschek G., FESTO - Experience with the ESSI Application Experiment ODB, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[16] Stewart J., Leonardo - The Networking Programme, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[17] von Neumann, Morgenstern, *Theory of Games and Economic Behaviour*, Princeton University Press, Princeton, New York, 1944

India: New Software Opportunities

M. D. Rao

Abstract

The Indian software industry provides the twin advantages of lower cost and high quality software development. Several international organisations have already reaped the benefits of this by outsourcing development projects to India, by setting up a joint venture with an Indian business group or by setting up their own subsidiary office. With the recent economic reforms taking hold, the Indian economy is undergoing a major transformation. The software industry has been growing at a rate of 50% over the last five years. The potential of the Indian industry and marketplace has not yet been tapped by European organisations. 3SE's *raison de etre* is to change that and be an enabler of increased alliances and joint ventures in software between European and Indian organisations.

The business environment in India

India has attracted widespread attention as an emerging market. With a population of about 900 million, India is the second most populous country and the fifth largest economy in the world by purchasing power parity. It is a country of immense, yet unrealised potential. This makes it one of the most exciting emerging markets in the world.

In addition, India is a stable parliamentary democracy with an established legal system, vibrant capital market and a mature financial system. The country has had a mixed economy: the government took responsibility to provide infrastructure, while the private sector flourished in other, diverse industry segments. English is widely spoken in the country, and is the business language throughout urban India.

Realising the need to accelerate growth, the government has been moderating its policy to provide increasing freedom of entry, investment, location and operation. Private - and importantly, overseas - participation and investment in core sectors is now welcome. Whereas a few years ago, the air traveller was obliged to fly the solitary domestic carrier Indian Airlines, he now has a choice of travelling another half-a-dozen private operators; automobile giants such as Daewoo, FIAT, Ford, General Motors, Mercedes Benz and Peugeot are busy implementing plans with Indian partners to catch up with Suzuki - the lone overseas player to operate in the Indian market till recently. Other global giants across industry sectors have also announced investment plans of hundreds of millions of dollars for their India operations. But it is the telecom industry that is creating the most excitement. The challenge and rewards of transforming the country's largely primitive and extremely limited facilities into a system on par with the best in the world, has seen giant telecom vendors the world over, without exception, establish a presence in India.
Many international entrants have eagerly leveraged their Indian connections to not only address emerging opportunities in India, but to source products and services to make them more competitive in other markets. India is attractive to international business because it is *not only a market, but also a partner for global competitive advantage*. Mr. Jack Welch, chairman of General Electric summarised this best when he said, in an interview to Business India,

"... We are going to see India as the intellectually most cost competitive country in our whole array of global businesses...We are going to be in every business that we are in the US, and we will invest heavily."

An overview of the Indian IT industry

As India integrates with the world economy, it is rapidly adopting the work culture of advanced countries: automation, and the use of information technology have ceased to be regarded as luxuries. The realisation that information and communications are mission critical has created a demand for state-of-the-art technology, that feeds upon itself.

Being relatively late in embracing information technology has had its advantages for India: the installed base of just over a million computers comprises mainly PC-compatibles and "open" work stations and servers. Major hardware vendors from all over the world - Compaq, Dell, Digital, Fujitsu, Hewlett Packard, IBM, Silicon Graphics, and Sun are entrenched in the market, and offer products just days after they are launched overseas. The rate of growth - expected to accelerate with the ongoing economic liberalisation - is already an impressive 40% over the past five years. Indeed, the low installed base of computers is a powerful indicator of the promise this industry holds.

Open platforms has encouraged the use of software popular in developed IT markets: Windows, NT and Unix are major operating system environments; Novell is the predominant networking software; the database pie is shared by Oracle, Informix, Ingres; and the same CASE tools, object oriented development software and other development environments in vogue in the west, are favourites here.

A look at the Indian software industry

An exploding domestic market

The past two years have seen the emergence of software as a major constituent of the Indian IT industry. As the chart below shows, the market has grown at the breathtaking rate of 50 % over the past five years.

The Indian domestic software market (Rs billions) CAGR 50%

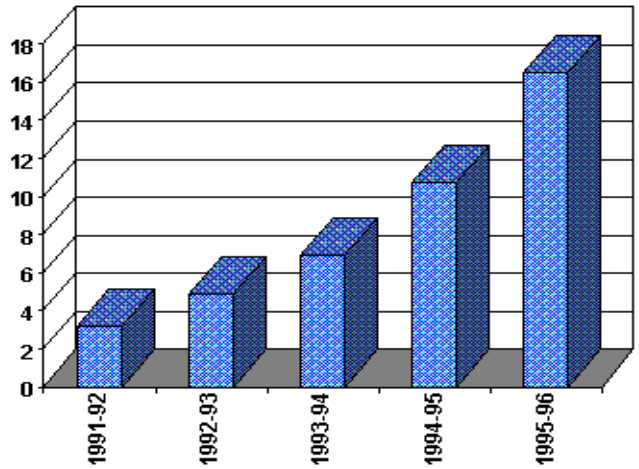


Fig. 1 Source: NASSCOM

Popular computing environments

Because India embraced information technology later than the industrialised world the computing environments popular in India are current, and "open". The following chart shows the software environments common in India.

Software Development Platforms (% companies surveyed with platform expertise)

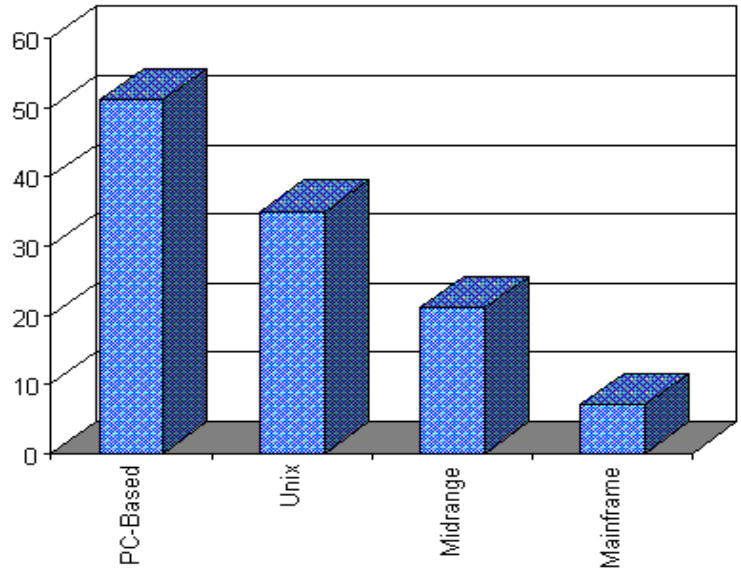


Fig. 2 Source: NASSCOM

Indian software companies have skills in diverse hardware platforms as the following table shows.

Popular hardware platforms (% companies surveyed with platform expertise)

% Companies	Hardware Platform
98	PC
84	LAN / Novell
83	UNIX
75	AS/400
73	IBM Mainframe
71	DEC
67	HP
62	Sun
57	Unisys
57	MAC
57	RS 6000
50	Tandem

Fig. 3 Source: NASSCOM

This compatibility in environments is a big advantage for international companies interested in the Indian software market - whether to promote their products, or to outsource products and services from here.

Major application segments

The manufacturing industry is presently the largest consumer of IT products. Of the other major application segments, the financial services segment shows great promise. The rapid ongoing modernisation of stock exchanges, banks (India has a network of over 60,000 branches, most of which use manual systems) and money markets augurs well for the sustained development of software solutions for this sector. Important market segments for software and services are listed below.

Important software segments (% companies surveyed in segment)

% of companies	Application segment
74	Banking

70	Manufacturing
70	Retail & Distribution
68	Communication
67	Transport
61	Government
55	Insurance
50	Hotels
50	Others
25	Defence

Fig. 4 Source: NASSCOM

Winning the confidence of clients overseas

It has become evident to companies all over the world that working with India helps create a competitive edge. While it accounts for a small share of the global software market, Indian software - mainly through services - is being increasingly used worldwide. The chart that follows endorses the growing popularity of Indian software:

Indian software exports (USD million) CAGR 35%

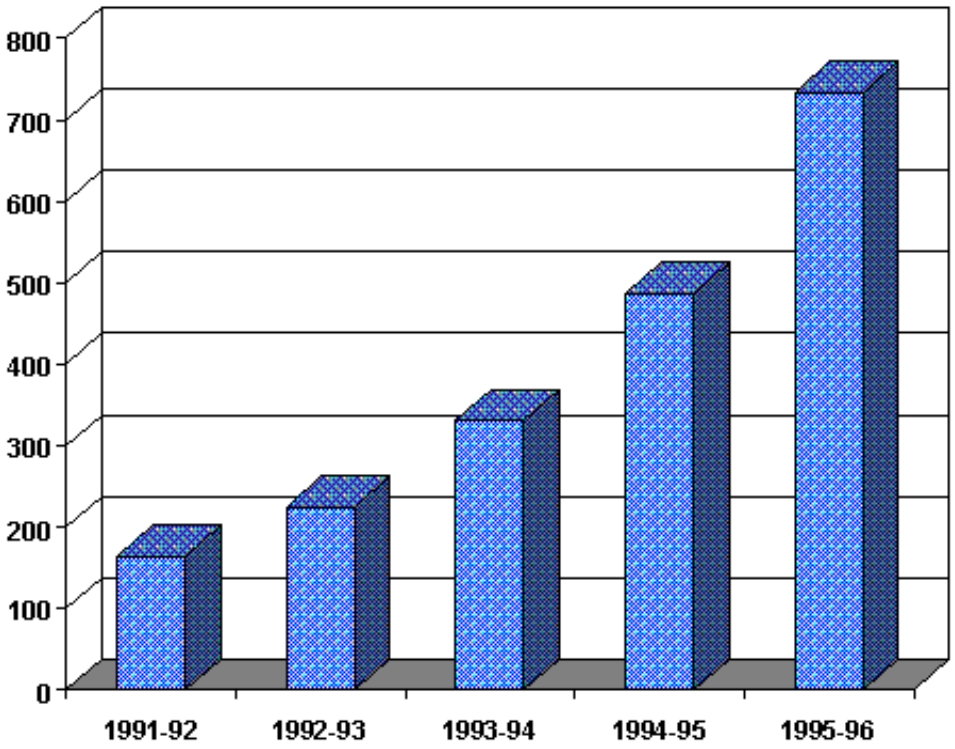


Fig. 5 Source: NASSCOM

Ed Yourdon, quoted in NASSCOM's 1996 Strategic Review observed, "the Indian software industry is now selling expertise in managing entire projects; the cost and identity of individual software engineers has become almost irrelevant."

This is borne out by the chart below: Indian companies now execute a significant part of their contracts from India, whereas as recently as in 1990, companies here were almost invariably just subcontractors for skilled personnel. Another noteworthy trend is the joint development in India of software packages for software product vendors overseas.

Software exports increasingly done from India (% 1990 & 1995)

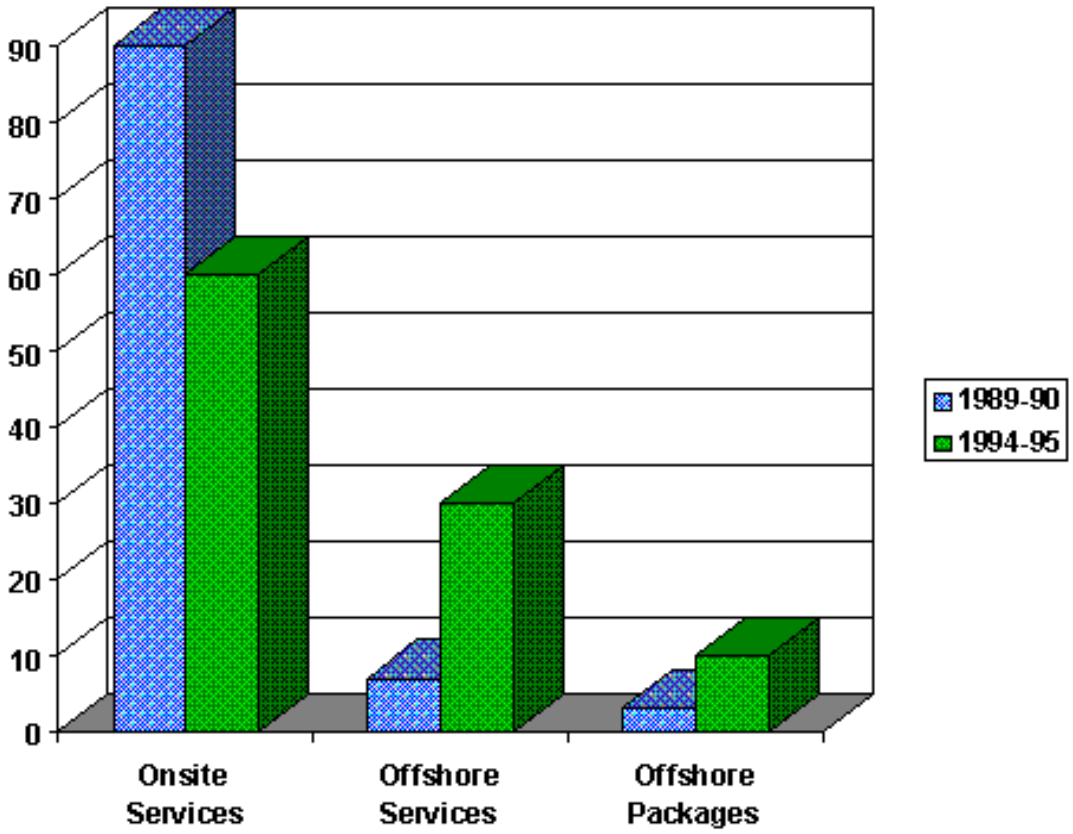


Fig. 6 Source: NASSCOM

This phenomenon has been made possible by several factors, including: the ready availability of a skilled workforce conversant with English, and with popular IT environments and project management; an emphasis on software and quality processes; major improvements in telecommunications infrastructure; and sustained government incentives for software units through lower duties, fiscal incentives and infrastructure.

Skilled Manpower

The education system in India is well developed. A string of technical institutions provide engineering education in various disciplines with specialised courses in computer and software engineering, computer applications, and related topics. The private sector plays a major role in supplementing the needs of the software industry. A host of training institutions have been established for developing expertise in computer programming languages, system analysis etc. Such institutions are given accreditation by the Department of Electronics, Government of India, to ensure quality education.
The National Association of Software and Service Companies (NASSCOM) has estimated that about 55,000 IT professionals, with varying skills, enter the workforce every year.

This system continuously reinforces a workforce of several thousand already experienced in executing projects worldwide, in software as diverse as those on old IBM mainframes to the latest in ASIC firmware development.

Quality in Indian software

The incentive for global organisations to look at India for software development is now no longer just lower cost; it is also high quality. A steadily growing sense of the importance of formal approaches to software engineering and software quality management pervades the Indian software industry. The industry is actively implementing systems to meet ISO 9000 requirements and to improve software processes.

The quality movement in the Indian software industry started in early 1993 with the ISO 9000 wave. Large numbers of software organisations embarked on this journey to improve quality systems. Today over 30 companies have been awarded ISO 9000 certification, and another 60 organisations are working towards it. It is likely that in the near future India will have the highest number of ISO 9000 certified organisations outside of the UK.

The movement got another impetus in late 1993 with the assessment of Motorola, India in Bangalore as an SEI-CMM level 5 organisation - a distinction achieved by very few organisations worldwide even today! Several Indian organisations, most of them of US origin, have since gone in for SEI-CMM either directly or over and above the ISO 9000 certification.

The Bootstrap method for software process assessment was initiated in India by an Indo-German project in 1991. Due to lack of sufficient marketing this did not catch on in the Indian market. Recently there have been some initiatives to revive this in India thereby providing an alternative to SEI-CMM for software process assessment.

There is considerable participation from Indian organisations in the SPICE project: 20 organisations have responded to the Call for Participation. (3SE is the Local Trials Co-ordinator for SPICE Phase 2 trials in India.)

Outsourcing: How India compares

The many advantages described have made India one of the most popular partners to outsource software services from. A 1992 study by Maxi/ Micro funded by the World Bank, of China, Hungary, India, Ireland, Israel, Mexico, the Philippines and Singapore rated India a very close second to Ireland. A weighted score for each country was arrived at by assigning an individual score to each of the following 12 parameters (weights of each parameter are given in parentheses): Segment expertise (6); Labour cost (4); Labour supply (4); Ease of business (3); Ease of visas (3); English speaking (3); Technical competence (3); Education & training (2); Government incentives (2); Security (2); Telecom infrastructure (2); Domestic market (1). The chart below indicates the relative positions of the countries considered:

Countrywise ranking of competing countries in software/ services exports

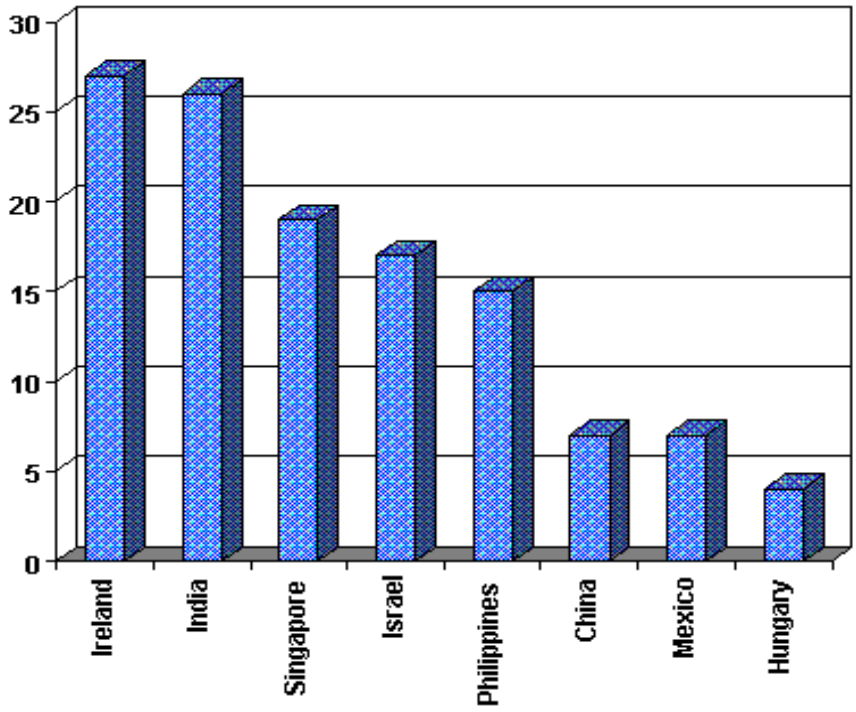


Fig. 7 (Source: World Bank report by Maxi / Micro Inc.)

In the years since then there has been a concerted drive to make the industry an even more attractive choice for outsourcing. The results are encouraging: in addition to the rate of growth, Indian software companies have been able to gain the confidence of their clients in their ability to undertake projects at home.

Capers Jones, in his paper *"International Software Benchmarking",* dated March 26, 1996 provides a comparison of the programmer costs per function point, and time-to-deliver for 1000 function points, between ten large software producing countries. As is seen in the charts below, India is a very attractive partner for companies seeking global competitiveness through software alliances.

International programmer cost comparison

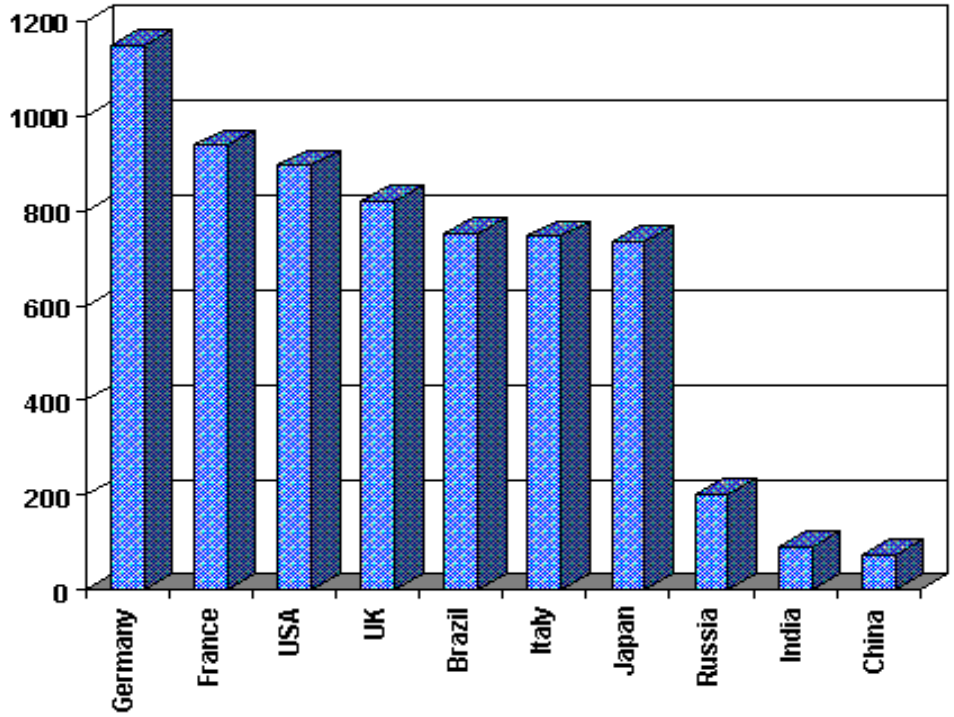


Fig. 8 Source: Capers Jones / SPR, 1996

International schedule comparison
(Lowest is best)

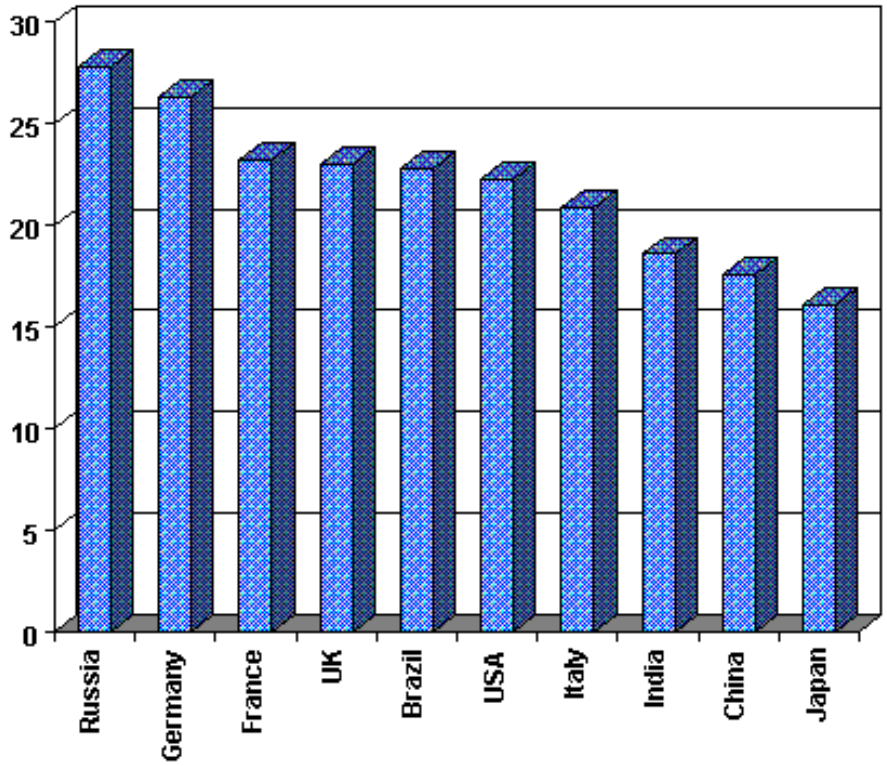


Fig. 9 Source: Capers Jones / SPR, 1996

Software transnationals in India

Most major global players in Information Technology have a significant software presence in India. Several transnationals are following suit for their requirements of in-house software services. Some of these are:

Some transnational companies with software operations in India

• AT&T	• Hughes
• Alcatel	• IBM
• Bellsouth	• Krupp
• British Aerospace	• Matra
• British Telecom	• Microsoft
• Bull	• Motorola
• Citicorp	• Olivetti
• Digital Equipment	• Philips
• Ericsson	• SGS Thomson
• Fujitsu	• Siemens
• Hewlett Packard	• Sprint
• Hughes	• Texas Instruments

Fig. 10

Paradigms for working with the Indian market

Different approaches have been used successfully by companies overseas to develop markets in India, and to outsource from India. Three types of operations are:

Through Indian companies on a contract basis

Companies such as Compaq, General Electric and Swiss Air have established offshore development centres in the facilities of large Indian software companies. These centres function as “virtual offices” - logical extensions of the development groups located in the US or Europe with teams of professionals dedicated to working for the requirements of a single client for its software requirements. A variation of this approach is for an international organisation to have an alliance with an Indian software organisation for a specific project. Using this approach work has been done in India for organisations such as London Underground, SNCF and Telesoft.

Through joint ventures with Indian companies

Financial partnerships with Indian business houses are a second alternative. Some successful joint ventures of this kind in the software industry are Tata Unisys, Dun and Bradstreet Satyam Software, Mahindra-British Telecom, BAe-HAL and Usha Matra.

By establishing their own office in India

Large organisations like Motorola, Origin (Philips), SGS Thomson and Siemens have set up wholly owned companies in India, and source their software for global markets from them. Smaller companies are following this trend: examples are the CAD software company Visionics of Sweden, and LEC in the financial services sector, of Denmark.

Major software centres in India

Bangalore in South India has come to be looked upon as India's "Silicon Valley". The city meets many of the chief considerations in the choice of a location for transnational companies: the availability of good technical manpower, proximity to quality educational and research institutions, telecom infrastructure, accessibility to overseas travellers, accessibility to the domestic market, the business and social environment, and climate. However, several other cities in India have much to offer companies looking to establish software operations in India: transnationals may be found in different Indian cities. The table below shows the distribution of the headquarters of the top 200 Indian software companies:

Locations of the top 200 software companies in India

City	Number of Companies
Bombay	68
Bangalore	56
Delhi	30
Hyderabad	16
Madras	15
Calcutta	8
Pune	7

Fig. 11 Source: NASSCOM

The Government of India has recognised the importance of the software industry in India, and is actively promoting it.

Making contemporary hardware and software available to the industry in India is critical to the use of IT in India, and for the development of the industry. Import duties have been progressively brought down - customs duties on software are now a mere 10%. With the major vendors competing keenly for a share of the burgeoning market, products are available in India almost as soon as they are announced anywhere else in the world.

The government also offers several fiscal incentives to software export units, such as tax holidays, and duty waivers. In addition, zones - of which software technology parks are an example, have been set up to provide computing and datacom infrastructure to software export units. In addition, procedures for setting up such operations has been simplified. The map below indicates the locations of software technology parks in India.

Software Technology Parks in India

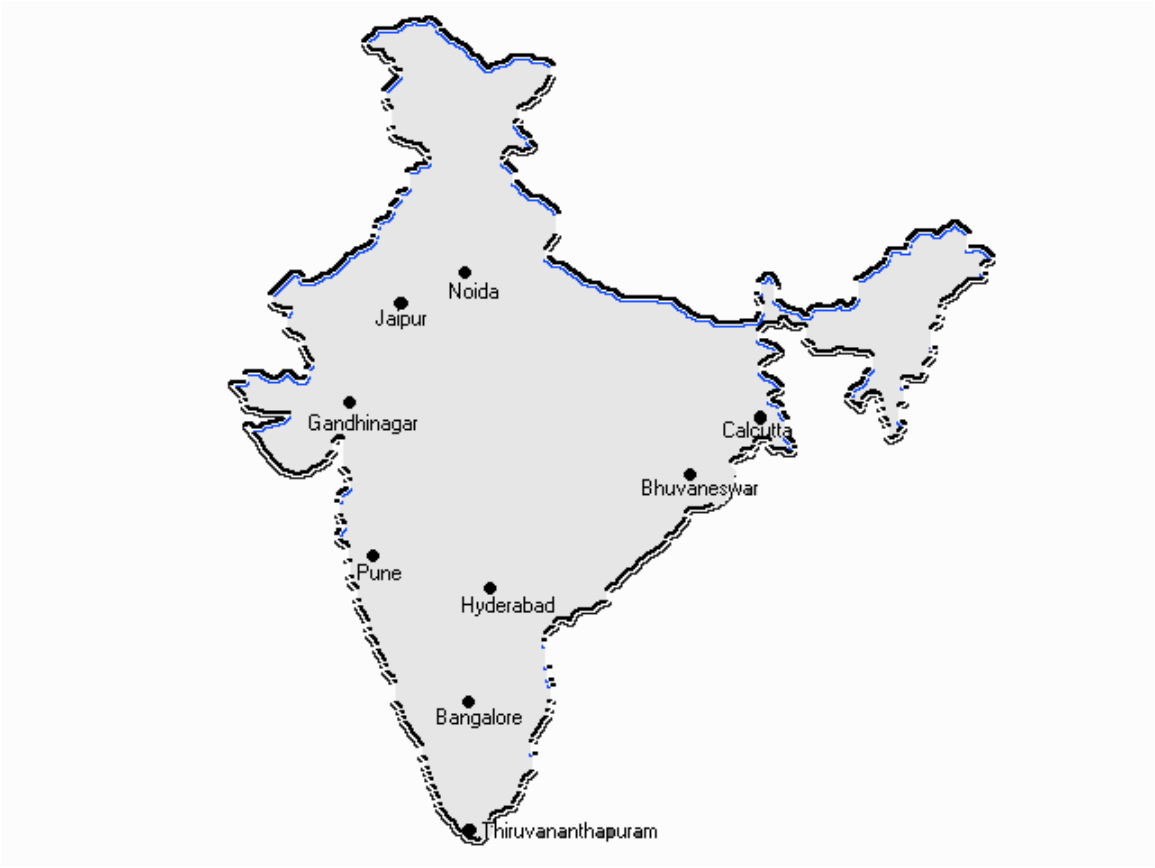


Fig. 12

3SE: Europe's Window to India

An enormous potential exists for cooperation between Europe and India in software, for the several reasons discussed. Opportunities can be harnessed in different ways: European organisations can increase competitiveness by outsourcing high quality, lower cost services from India; explore new markets in India and neighbouring countries; collaborate on projects and products for markets in third countries; outsource projects in India - such as Year2000 software conversions - that would otherwise consume in-house resources that could be better utilised on other tasks.

3SE has been established by the European Commission and the Government of India to promote such cooperation between the EU and India in the field of computer software.
3SE helps European organisations to:

- identify appropriate Indian companies to outsource software from
- locate partners to jointly address business opportunities in third country markets, whether for products or services
- find the right distributor for their products in India
- source Indian software products for distribution in Europe and globally
- help establish operations in India by providing information on local laws and regulations, business practices, and the like. 3SE also assists companies with legal procedures and in obtaining clearances

Companies of diverse sizes in Austria, Belgium, Germany, Italy, the Netherlands, Sweden and the United Kingdom have used these services to advantage. Recent examples are: a British company that offers solutions in Geographical Information Systems that is negotiating a business arrangement for software outsourcing with a company we identified for them; an Italian bank that finalised a joint venture with a company we had shortlisted for them to initially develop software for in-house, and later for commercial purposes; and a large German conglomerate establishing a software research subsidiary, for whom we have helped incorporate the company in India, obtain government clearances, locate and lease office and residential accommodation, coordinate recruitment, and identify suppliers of equipment.

M.D.Rao may be contacted at:
3SE, 8th Floor, DJCC, Hudson Circle
Bangalore 560 027, India
Ph: ++91.80.2211143 Fax: ++91.80.2211152
email: mdr@3seblr.soft.net URL: http://www.3seblr.soft.net

References

1. *Business India*. Bombay: Business India. 1995 January.

2. NASSCOM. *The Software Industry in India 1996*. New Delhi: NASSCOM; 1996.

3. Maxi/ Micro Inc. *India's Software & Services Export Potential & Strategies.* New Jersey: July 1992. Vol1, 55.

4. Capers Jones. *"International Software Benchmarking".* USA: SPR Inc.; 1996 March.

Software Process Engineering Activities in Québec

Claude Y. Laporte, CD, M.S., M.S.A., Oerlikon Aerospace, cylaporte@oerlikon.ca

Claude Y. Laporte obtained in 1973 a first degree diploma in physics and mathematics at Collège militaire royal de Saint-Jean. He was also sponsored by the Department of National Defense to pursue graduate studies in science. In 1980, he obtained a master in physics at Université de Montréal, then, in 1986, a master in Applied Sciences from the Department of electrical and computer engineering at École Polytechnique de Montréal. He was an officer in the Canadian Armed Forces and a professor for over 10 years at Collège militaire royal de Saint-Jean. From 1988 to 1992, he was involved in the feasibility study that led to the implementation of the Applied Software Engineering Centre. He left the Canadian Forces in 1992 at the rank of major. Since then, he has joined Oerlikon Aerospace where he co-ordinates software and systems process engineering activities. He is the president of the Montreal Software Process Improvement Network (SPIN).

Abstract

This paper is divided in three parts. The first part will present the Applied Software Engineering Centre, its history, its mission, and the services offered. The second part will present a brief profile of organisations that have undertaken to improve software processes utilising mainly the Capability Maturity Model developed by the Carnegie Mellon University Software Engineering Institute. The third part will present lessons learned in process improvement. This paper is an update of a presentation given on the occasion of a workshop held at GMD, a German software research centre. (Laporte 1993, 1995, 1996a).

Key Words

Software Engineering, Applied Software Engineering Centre, Capability Maturity Model, Software Engineering Institute, Trillium, Management of Changes. Software Process Improvement Network, Camélia.

1 Introduction.

We often hear of problems in software intensive systems. Typically the problems are: systems that do not meet customer's requirements, unreliable operation, costly development and maintenance and unmet development schedule and budget. The U.S. Department of Defense reports (DoD 1987) that after two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and technologies (e.g. tools), industry and government organisations are realising that their fundamental problem is the inability to manage the software process. In this paper we will present software process improvement initiatives undertaken by private and public organisations of Québec.

2 The Applied Software Engineering Centre.

The Applied Software Engineering Centre (ASEC) was created as a result of an agreement between the Computer Research Institute of Montréal (CRIM) and six Canadian corporations committed active in the development and maintenance of software for critical applications: Bombardier, CAE Electronics, Keops Informatique, Lockheed Martin, Oerlikon Aerospace and Spar Aerospace.

An action had been undertaken in 1988 in the form of a feasibility study financed by 13 companies and the federal and Québec governments, with the participation of the Collège militaire de Saint Jean, which confirmed the role and importance of software engineering in improving the productivity and competitiveness of Canada's industry.

Encouraged by these results, the study's sponsors decided in 1990 to draw up a business plan aimed at creating a software engineering centre, the mission of which would be to assume a leadership role the technological level and to assist industry, where such an expertise is required, to improve their competencies in software engineering. In 1991, the Applied Software Engineering Centre became a division of CRIM.

ASEC was created to respond to an urgent need expressed by the industry in Canada, which is facing a challenge the outcome of which will be decisive. Although information technologies have become an overriding factor of productivity and innovation in all sectors of activity and although demand for more and more complex software has increased in a spectacular way, the lags in terms of software development as well as the lack of qualified personnel are seriously hampering our industry's progression. In this matter, cost overruns, schedule slips, lack of product friability and system failure due to software bugs are innumerable. Even worse, in certain critical applications, these problems can have serious repercussions on public security or result in significant financial or social losses.

The mission entrusted to the Applied Software Engineering Centre is to provide access to and training in the best software engineering managerial and technical solutions. Its target clients comprises companies and agencies that rely on information technology to improve the productivity and quality of their services and products. ASEC offers four main categories of services: services related to software engineering process such as software process assessment, auditing of suppliers' competencies and advising, training, awareness to new technologies by means of appropriate activities, as well as implementation of and relevant support to specific interest groups. ASEC is also part of a network of similar centres subsidised by the federal government.

ASEC signed in December 1995 a co-operation and research agreement with the Software Engineering Institute (SEI) of Carnegie Mellon University. In accordance with this first SEI's international agreement, ASEC is not only able to utilise the SEI's assessment methods to assess the maturity of the software process engineering, but also transfer to industry in a more efficiently way methods and technics permitting to improve software development and maintenance practices.

Until now, the "Capability Maturity Model" (CMM) has only existed in English, which limited considerably its usage for the French-speaking community. Fortified by its strategic agreement with the SEI, ASEC jointly with organisations from France (CEGELEC, Dassault Électronique, the French Department of Defence, Snecma Elecma and Thomson-CSF) and other from Quebec (Bombardier and Hydro-Québec) as well as the federal and Quebec governments (respectively Industry Canada and ministère de l'Industrie, du Commerce, de la Science et de la Technologie) the translation into French of the Capability Maturity Model developed by the SEI. ASEC also participates in the creation of software Web site in French. This Web site will comprise not only French translations but also information conceived and circulated in French through all French-speaking communities.

3 Software Capability Models developed in Québec.

Since 1982 (Coallier 1995), Bell Canada has also been developing a Software Capability Maturity Model to assess the processes of its telecommunication systems suppliers in view of reducing risks. Trillium is now part of the management program of Bell Canada's suppliers. Trillium insists on the self-improvement of software manufacturing processes as an approach allowing to improve the quality and reliability of telecommunication systems and reduce their operation and maintenance costs. This is critical when considering that Bell Canada's telecommunication network depends on more than fifty million lines of code.

Trillium was developed by Bell Canada, Nortel and Bell Northern Research. Although strongly inspired by the CMM Model, several requirements were drawn from the ISO, Bellcore, IEEE standards as well as from the criteria related to the Malcom Baldrige National Quality Award. A major difference between the CMM and Trillium is that the latter contains key process areas which vary on a five-level scale (road map) contrarily to CMM where each key process area lies at one capability level only. The Trillium model also comprises practices that are not covered in the CMM.

A France-Québec project, started in 1992, deals with the adaptation of the Trillium model and with the creation of an evaluation method based on the CBA-IPI method for use, namely, in the information software sector. The project comprises mainly the translation in French of the Trillium model, the addition of practices related to the information systems sector, and the change of terms in order to be compatible as much as possible with the ISO 12207 Software Life Cycle Process standard. The result is named the Trillium-Camélia model. Some domains, road maps, and practices have been added to cover more extensively the development, maintenance and operation of information systems. They are: business process re-engineering, architectures, financial life-cycle analysis, data management, product re-engineering, and operations. An evaluation method was created and embedded in a 3 to 5 day course. The method is named Camélia. Within this method, a questionnaire of more than 100 questions, based on the trillium-Camélia model, has been created as a tool to have a first overview of the maturity of the organisation evaluated. The Trillium-Camélia model was tested both in Québec and in France, in 1995 and in 1996. It should be published soon.

4 First Experiments with the Maturity Model

A first exposure to the software process assessment methodology developed by the Software Engineering Institute (SEI) was done in Montréal in the summer of 1989. Two members of the technical staff of the SEI conducted a one-day workshop at École Polytechnique, Montreal. The workshop was attended by 50 persons. The participants came mainly from defence, aerospace and finance organisations, of both the private and public sectors. During the workshop, the participants answered the SEI questionnaire, that was used to conduct formal assessments (Humphrey 1987). The questionnaires were compiled, and the results were that 93% of the participants to this workshop worked for organisations at the initial maturity level (level 1) and the remaining 7% were at the repeatable level (level 2) of the maturity scale. Although the assessment of organisations according to the SEI's approach would have been far more stringent, these results remain nevertheless indicative of the situation prevailing at that time.

As a comparison, the United States conducted similar workshops and gathered data from 113 projects (Humphrey 1989). The assessment workshop results as of January 15, 1989, indicate that the majority (86%) of the participants reported projects at the initial level (level 1). Fourteen per cent (14%) of the participants reported projects at the repeatable level (level 2) and one per cent (1%) reported projects at the defined level (level 3).

Following the tutorial held at École Polytechnique, some organisations decided to conduct software process assessments and improvement activities. The following section will present organisations that have performed software process assessments and improvement activities.

5 Some Software Process Improvement Experiences in Québec

The data published here have been supplied by the organisations themselves and not by ASEC, since the latter has to respect the confidentiality of the assessments done by the organisations. Moreover, we will only be discussing the organisations that have undertaken the improvement of their processes utilising either the CMM, a model, or an assessment method associated to the software capability maturity model such as Trillium. Because of space limitations, process improvement activities related to the ISO 9000 standard will not be discussed.

5.1 CAE Electronics - Fighter Aircraft Maintenance Group

In 1990, CAE Electronics, in collaboration with Bombardier, decided to go ahead in performing a Software Process Assessment using the SEI's assessment method. A group of CAE Electronics is responsible for the maintenance of the software of the Canadian Armed Force's fighter aircraft CF-18 fleet. For this assessment, it was decided that the assessment team would be composed of representatives from the customer's organisation as well as representatives from the assessed organisation. The on-site assessment was performed in February of 1991 and the action plan was published in September. The costs of process assessment and improvement activities (Lambert 1992) are summarised below (Table 1). This division has also performed, in collaboration with ASEC staff, in the summer of 1994, an assessment using the new method developed by the SEI. This method is called CBA IPI (Capability Maturity Model - Based Appraisal Internal Process Improvement). We know that the group responsible for the maintenance of the fighter aircrafts was assessed at maturity level 2, hence mastering all objectives of the 6 key sectors of the CMM. The assessment has also showed that several objectives of level 3 were reached.

Assessment training and consulting cost:		Cdn$40,000
Labour:		
	Training	160 hours
	On-site assessment	240 hours
	Action plan elaboration	500 hours
	Action plan implementation	2,500 hours

Table 1: Assessment and Improvement Costs

5.2 Lockheed Martin Canada

In 1991, Lockheed Martin Canada, formerly known as Paramax Systems Canada, decided to perform an SEI assessment. Lockheed Martin Canada is an organisation mainly responsible for the development of the Canadian patrol frigate's computer system. The 2 million source lines of code software were developed by a large team of over 200 engineers, geographically dispersed in Canada and in the United States. Since 1991 Lockheed Martin Canada has been improving its processes using the SEl's CMM, TQM (Total Quality Management) and ISO 9000 principles.

5.3 Hydro-Québec - Automatisation Group

In 1993, four organisations performed SEI assessments. The first organisation is the province of Québec's electricity supplier: Hydro-Québec. Its automatisation department conducted an in-house assessment using the SEI questionnaire (Humphrey 1987). This department, staffed with 17 people at that time, is mainly responsible for the development and maintenance of real-time embedded software that controls the Quebec's electrical network.

5.4 Oerlikon Aerospace

The second organisation that conducted an assessment in 1993 is Oerlikon Aerospace (Laporte 1996b). This organisation is responsible for the production of an air-defence anti-tank system. The software engineering department, staffed with over 25 people, is responsible for the maintenance of the weapon's software; the command control and communication system's software; simulation software and instrumentation software. The on-site assessment was done in collaboration with the customer and the Applied Software Engineering Centre, in the spring of 1993. The action plan was completed in December 1993 and the process improvement activities were initiated in January 1994. The action plan aims at implementing within Oerlikon Aerospace level 2 and 3 practices in compliance with the SEI's model. The organisation is

planning a re-assessment, in collaboration with the Applied Software Engineering Centre, in 1996.

Oerlikon Aerospace has also undertaken, in 1995, a systems engineering improvement program. The effort was started by performing an internal assessment using the Systems Engineering Capability Maturity Model (SE-CMM) (Bate et al., 1995) and the SE-CMM Appraisal Method (SAM). A beta version of the systems engineering process has been defined, and pilot projects are being conducted. As pilot projects are using the new process, practices are identified and incorporated into the process description. A parallel effort is also conducted to integrate the systems engineering process to the in-use software engineering process.

5.5 Montréal Trust (Scotia Bank)

The third organisation that performed an assessment is the Montréal Trust. Montreal Trust has been, since then, acquired by the Scotia Bank. Montreal Trust used to offer a range of financial and trust services. It administered assets of $64 billion. The on-site assessment was done in spring of 1993 and the recommendations were presented to management in fall of 1993. Montréal Trust was assessed as a strong level 2 and was expected to reach level 3 by the end of 1994.

5.6 CAE Electronics - Energy Control Department

CAE Electronics is the fourth organisation that performed an assessment in 1993. CAE Electronics mainly develops and manufactures a wide range of military and civilian simulators. In September, the Energy Control System Department, staffed with 90 software engineers, performed an assessment of its processes in collaboration with a customer. CAE uses the ISO 9000 standard as an objective and the CMM as a guide to implement practices compliant to the ISO standard.

5.7 Hydro-Québec - Research Institute

The management responsible for the Network Technologies (DTR) of Hydro-Quebec's research institute (IREQ) has undertaken to improve its processes in 1993 (Lafleur-Tighe 1996). This initiative follows the basics of several development models, particularly the CMM. At IREQ's, the improvement is done by establishing methodological guides, such as definition of the requirements, the development plan and the typical mandate, related to software engineering and system engineering fields. By the end of 1996, the DTR should ensure a repeatable development process and be able to supply process descriptions and/or documentary standards for each step of development, as well as umbrella activities in planning and project-tracking, configuration management and quality-assurance support. It is also foreseen to perform an assessment of the processes in 1996 and a follow-through assessment

in 1998. The DTR's objective is having a defined process, i.e. a level 3 according to the CMM, by 1999.

5.8 IST Group

In 1994, the IST Group started a process improvement initiative using the S:PRIME assessment method (this method is described further in this text). This initiative began by a training session in 1994, followed by a series of assessments in 1995 in Toronto, Quebec and Montreal. An action plan was approved in May 1995. The initiative permitted to identify the best practices, to complete their descriptions and transfer them in other sectors. Each sector could customise the practice to its own requirements. One of the objectives aims at obtaining the ISO certification in 1996.

5.9 Ericsson's Total Business Improvement Program

In 1994, the company Ericsson undertook an improvement program (Modafferi 1996). The ISO certifications had been obtained in 1993. The initiative followed a reflection on the challenges to be faced by companies world-wide. Following this reflection, it was decided that the software capabilities were among the company's major objectives. In May 1995, an assessment was realised in Montreal by a team of experts belonging to the mother company. It is interesting to underscore here that Ericsson conducted over twenty assessments on its various sites. The assessment method used was very similar to CBA IPI. Elements were added to it, from the assessment method called "European Quality Award" in order to add practices that were not covered by the method CBA IPI. The company foresees to conduct a second CBA IPI assessment in 1997, it will be using the S:PRIME method to assess the progress made between two major assessments.

5.10 Canadian Marconi Company

Canadian Marconi Company (CMC) has a wide range of domains and applications, and the domains are organised in business units distributed over a number of sites in Canada and in the United States (Sayegh 1996). The objectives of the software process effort are: to address the needs of all business units; to ensure buy-in from all entities; and to optimise cost effectiveness. At CMC, software process improvement is managed as a project and a management steering group provides oversight and verifies the progress of the effort. A process improvement project is started by performing a CBA IPI assessment with accredited SEI assessors and establishing a software engineering process group. A software process has been defined with a minimum set of requirements addressing the needs of the business units. Each business unit tailors the process by adding practices as required. CMC terminology, instead of CMM's terminology, has been used such that processes are easy to use and unambiguous.

In 1994, Canadian Marconi Company initiated its process improvement program. A first CBA IPI assessment was performed, at the Montréal site, by the Applied Software Engineering Centre. An improvement plan was developed and approved in April 1995.

5.11 Régie de l'assurance -maladie du Québec

In 1996, the Management Information System (MIS) department internal to the Régie de l'assurance-maladie du Québec (RAMQ) decided to initiate an improvement program by using the result of the Camélia project. The effort started by performing an assessment (Bistodeau 1996). Two assessments were conducted under the supervision of ASEC and the Treasury Board of the Québec government: one for the development and the maintenance processes of the information systems, and one for the operation processes. The first assessment evaluated about 300 practices related to development and maintenance while the second evaluated about 150 practices related to the operations of information systems. The action plan based on these two assessments should be completed by the end of October 1996. This action plan will be inserted in the overall enhancement plan inside the MIS department.

5.12 Bombardier - Mass Transit Division

In previous train systems, sub-systems were controlled through electro-mechanical devices. They were developed and tested individually and then integrated on the railway car. Today, not only sub-systems are more complex, but they are controlled by software and they often communicate between each other. Moreover, once defined, requirements are often modified. This has led the mass transit division to define a software development process (Bélanger 1996). In addition, since many components are acquired through suppliers, subcontracting management practices were defined. An assessment was also performed using the S:PRIME method.

6 Process Related Activities

6.1 Montréal SPIN

Montréal is the host of a SPIN (Software Process Improvement Network). Essentially, a SPIN is an interest group composed of software professionals from industry, government, academia, professional organisations, and consulting agencies. The SPIN provides a forum for the free and open exchange of information on software process improvement. The SEI provides some support to the SPIN (Marchok). In fact, the SPIN in Montréal is part of an international network of interest groups called "SPIN for Software Process Improvement Network". The 1996 SPIN directory listed 42 U.S. and 29 international SPIN organisations. The Montréal SPIN was founded in 1993. Its mission is to facilitate the understanding, the adoption and the deployment of proven or innovation solutions for software process improvement. Each year,

the SPIN organises events such as tutorials, workshops and round tables. The SPIN is affiliated to the Applied Software Engineering Centre; the meetings are generally held at ASEC facilities. In addition, the SPIN benefits from the administrative services offered by ASEC (e.g. mailing, reservation, accounting).

The co-operation between the Montreal-SPIN, ASEC, the SEI and the International Council on Systems Engineering (INCOSE) gave rise to an international symposium on systems and software process improvement, entitled Vision96, was held in Montreal in October 1996. This symposium was aimed at gathering managers, professionals and contributors intervening in the continuous implementation and improvement of systems and software processes. It represented a unique opportunity to perfect participants' knowledge and enrich their vision by sharing their experience and concerns on subjects such as investing, stakes, risks, profits and international trends in process improvement. Over 238 persons from 10 countries attended the symposium.

6.2 Software Engineering Standards Interest Group

ASEC also hosts an interest group that focuses on software engineering standards (GINIGL). More specifically, this group is very active in the ISO-SPICE project (lnternational Standards Organisation: Software Process lmprovement and Capability Determination (Paulk 1994b). In collaboration with the interest group, ASEC participated to the first field trials of this forthcoming ISO standard, in 1995. More than 35 international organisations participated in these field trials, of which one took place in Quebec. Hydro-Québec's Automatisation Department, i.e. 35 people, participated to the field trials. An action plan was developed following the assessment: it integrates both the concepts of the SPICE model and those of the SE-CMM model (Systems Engineering CMM). The second SPICE field of trials will begin in May 1996 and will last 12 months. Again, the GINIGL and ASEC will play a major role in the co-ordination of the field of trials in Canada, Central America and South America.

6.3 S:PRIME Assessment Method

Since there is close to 500 small or medium businesses that develop software in Québec, it was felt that these organisations could not afford the resources of performing a CBA IPI assessment and still be able to set aside resources needed to address the findings of the assessment. A CBA IPI typically requires around 1500 person-hours on the part of the assessed organisation. Also, an organisation that do not have in-house assessors must add the cost of a certified assessor who will spend at least ten days in the preparation and the conduct of the assessment. Therefore ASEC, in collaboration with industrial partners, developed a risk evaluation method based essentially on the CMM key process areas. The method is called Software: Process Risk ldentification Mapping and Evaluation (S:PRIME). The result of

S:PRIME assessment consists in an identification of the risks the organisation or the project are faced with, as well as in an identification of the CMM practices that should be improved or introduced in the organisation or project in order to prevent these risks (Poulin 1996). The method typically takes 100 staff-hours to perform the assessment of an organisation. Once an organisation has been trained, it can perform by itself follow-up S:PRIME assessments in order to track action plan progression or identify other areas of priority.

The method consists in administering two questionnaires. A first questionnaire is answered by managers in order to identify their perception of the level of risk in their project(s). Seven risk categories are addressed. They are risks related to: requirements, design and production, the development environment, the development process, management, personnel, and external constraints. The taxonomy of these risks constitutes the result of the work performed by the SEI these past years. A second questionnaire is answered by practitioners assigned to the assessed project(s). This questionnaire addresses level 2 and 3 key process areas of the CMM augmented with two practices: customer service and organisation culture. An algorithm computes the expected value of the risk level for each risk category and each practice area. Figure 1 illustrates graphical results generated by the software tool.

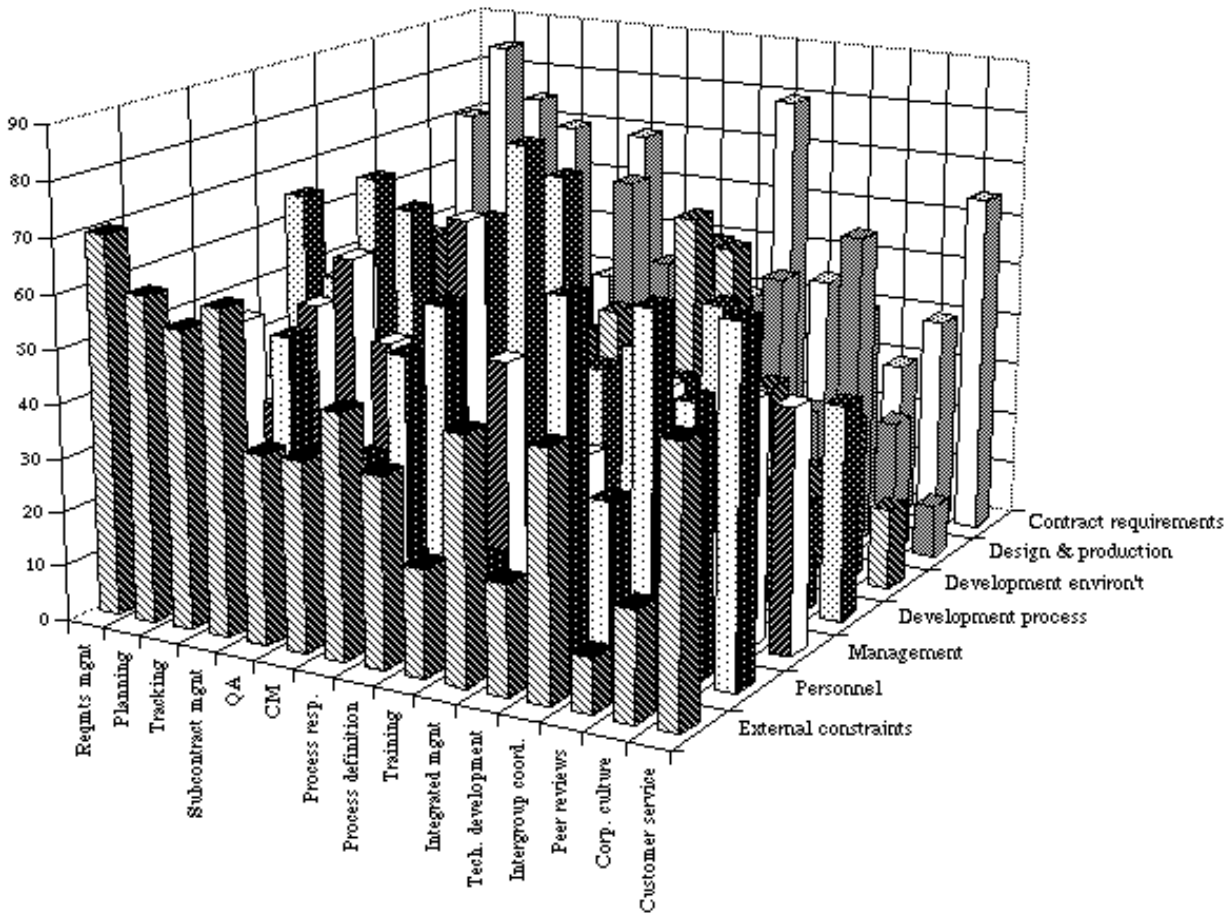


Figure 1. Typical results of a S:PRIME assessment

So far, twenty two S:PRIME assessments have been performed of which two in Chile and one in France. The method has also been translated into French and Spanish. The method is supported by a software tool in order to facilitate the capture, the analysis and the presentation of the date gathered during an assessment. An action planning approach also complements the S:PRIME assessment.

6.4 Personal Software Process

The Personal Software Process (PSP) is a framework for doing disciplined software engineering. The PSP was developed under the direction of Watts Humphrey (HUMPHREY 1994, 1996) of the SEI. The PSP consists in activities similar to several key sectors of the CMM. Essentially, PSP shows professionals how to use measurements and statistical methods to plan and control their work. It also helps them to make accurate plans, to estimate the accuracy of these plans, and to track their performance. They learn to define, evaluate and improve a software process that is tailored to their own evolving personal needs. This helps them to evaluate and progressively improve their own performance. CAE Electronics, in collaboration with McGill University, undertook a pilot study to see if the PSP could be adapted to their organisation (Shostak 1996). Twenty eight volunteers participated in the study. The approach was to provide the PSP lectures and then allow the volunteers to apply the techniques in their job.

6.5 Risk Assessment for Investment Decisions

An organisation, Telsoft Ventures Inc., with a software venture capital of $78.2 million uses a process maturity assessment as one indicator of risk level before making substantial investments in organisations (Mayrand 1996). Other issues evaluated are: financial health, technology created, market, technology and product maturity, and management maturity. A first process maturity is performed before a decision is made to invest in the target organisation. Then, once the investment is made a detailed process assessment is conducted, and an improvement plan is defined and executed. A joint assessment, based on Trillium, is performed. One of the goals, in performing a joint assessment, is to train the employees of the organisation. The improvement plan usually starts by documenting actual processes. Then, support and requirement processes and customer interfaces are defined. Finally, the development process is formalised and formal reviews are introduced. Re-assessments are conducted initially at 6-month intervals.

Table 2 lists organisations known by the author, that are actively involved in software process engineering activities. So far, most assessments were performed by large organisations, using

the SEl's approach. ASEC performed at least five SEI assessments since April 1994 and expects to conduct another five in 1996-97. Since in Québec the number of small and medium organisations outnumbers the number of large organisations, we expect a growing use of S:PRIME method. Finally, since it is expected that SPICE will become and ISO standard in 1998, it is possible that organisations choose to wait two or three years before deciding whether to adopt this type of assessment or stay with the SEI's approach. It is also possible that the SEI decides to map its maturity model to the SPICE framework. It is worth mentioning that the SEI is collaborating to the development of a System Engineering Capability Maturity Model (SE-CMM). This CMM is using a framework nearly identical to the SPICE framework for the mapping of maturity levels (Bate et al., 1995).

Organisation	**Sector**	**Year**	**Activity**
CAE Electronics and Bombardier	Defence	1991	SEI - SPA (1)
Lockheed Martin Canada	Defence	1991	SEI - SPA (1)
Hydro-Québec	Utility	1993	Internal assessment using CMM
Oerlikon Aerospace	Defence	1993	SEI - SPA (1)
Scotia Bank (Montréal-Trust)	Finance	1993	SEI - SPA (1)
CAE Electronics	Energy Management	1993	SEI - SPA (1)
Hydro-Québec- IREQ	Utility - Research	1994	Internal assessment using CMM
Ericsson	Telecommuni-cations	1994	SEI - CBA IPI (3)
CAE Electronics and Bombardier	Defence	1994	SEI - CBA IPI (4)
Canadian Marconi Company	Defence	1994	SEI - CBA IPI (4)
M3i	Network Management	1994	S:PRIME (5)
Hydro-Québec	Utility Automatism	1995	SPICE
IST Group	Information Systems	1995	S:PRIME
Bombardier-Mass Transit Division	Transport	1995	S:PRIME
CRIM	Research & Development	1995	S:PRIME
RAMQ	Information Systems	1996	Camélia (6)

Note: 1. SEI - SPA: Software Engineering Institute Software Process Assessment with third party.

2. Internal assessment using CMM conducted without participation of third party.

3. SEI - CBA IPI: SEI - CMM based-assessment: Internal Process Improvement with third party together with additional practices.
4. SEI - CBA IPI: SEI - CMM based-assessment: Internal Process Improvement with third party.
5. S:PRIME: Software Process Risks Identification, Mapping and Evaluation
6. Camélia: Based on Trillium with practices for Management Information Systems

Table 2: Software Process Activities in Québec

6.5 Software Engineering Management Research Laboratory

This laboratory is located at l'Université du Québec à Montréal and directed by professor Alain Abran. Its mission is to develop, for our software engineering community, the analytical models and measurements instruments to enable them to improve their decision-making processes in order to meet their business objectives. The laboratory is funded partly by Bell Canada and the National Research Council of Canada. One field of research is the development of an evaluation and improvement model for software maintenance processes (Zitouni 1996). The model is largely inspired by the CMM for software. Since the CMM is heavily development-oriented, it does not necessarily apply to maintenance. The project will identify, describe, structure and model the components of the proposed model and insert them in a CMM-like structure. The present version of the model is composed of 21 key process areas, 63 goals and 312 practices spread from level 2 to level 5. It also includes a glossary of 112 words specific to the maintenance domain.

7 Lessons Learned

These assessments enable us to learn certain lessons likely to be used by other organisations or companies in the future.

Lesson 1: Set Realistic Expectations for Senior Management

Appropriate expectations must be set prior to embarking on a process improvement journey. The trap consisting in communicating to management the idea that the initiative will be easy, fast and inexpensive has to be avoided at all costs. As a first step, a top management member realises the benefit that attaining a maturity level can represent for his organisation's competitiveness. As second step, a project manager or an external consultant states, in order not to upset the top management, that this objective is easily attainable. As a third step, top

management gives managers the mandate to attain this objective in a very short lapse of time. During the assessment, the managers face countless a string of findings. Findings that had been known by developers for a long time, but remained ignored due to the mode of management that consists in dealing continuously with the problems created (i.e. fighting fires), in a clumsy way at times, by managers. Top management, that had maybe already announced its objective to its peers from other organisations, realises suddenly that this objective will take a lot more time and resources than what had been estimated. At that time, three reactions are possible. Top management may accept the findings and confirm that it will continue to support the objectives announced. It may announce discreetly that it will be lowering its objectives. Finally, it can deny everything and renounce to implement an action plan to correct the deficiencies highlighted by the assessment. This decision could have a destructive effect on developers, since they know for a fact that the deficiencies they had been deploring for a long time are now known by everybody and will remain ignored for a long time.

The lesson to be remembered is to prepare a first action plan -- some sort of a brief appraisal of the situation status -- preferably by someone who is not involved in the sector targeted and to assess the time and resources necessary to assessing and, writing and implementing the action plan. One has to remember top management does not like bad surprises. Moreover, it is better not to proceed to an assessment if it is not intended to deal with the findings. As a matter of fact, once the problems are identified and publicised within the organisation, if the management decides not to act, it then sends a very bad message to practitioners.

Lesson 2: Secure Management Support

A second lesson for CMM level 1 organisations consists in realising that the assessment findings target the deficiencies of project management processes. It is necessary to create an environment where the management is ready to invest in the implementation of processes rather than blame its managers; in other words “where the management is ready to fix the process, not the people”. This is one of the reasons why it is necessary to also keep informed senior management representatives so that they can show understanding and full commitment when these findings are publicised within the organisation.

Beside senior management buy-in, it is essential that middle management and first line managers become champions of the process improvement program. The developers must receive very clear signals announcing that the changes advertised will be implemented and that they themselves will have to adopt new practices.

Lesson 3: Establish a Software Process Engineering Group

The Software Capability Maturity Model suggests the formation of a Software Engineering Process Group (SEPG) for any organisation heading toward level 3 (Fowler, 1990). Even for a level 1 organisation, it would be better that a small number of persons becomes active in process activities a couple of months before the on-site assessment. The SEPG should take this time to familiarise itself with the Capability Maturity Model and associated process improvement methods and tools. Ideally, in a large organisation, there should be one full-time person on the SEPG while the other members could be assigned on a part-time basis. Beside their technical competencies, the members of the SEPG should be selected based on their enthusiasm for improvement and the respect they have within the organisation.

Lesson 4 : Start Improvement Activities soon after an Assessment

With regards to the development of the action plan, the organisation should capitalise on the momentum gained during the assessment period. The organisation does not have to wait for a completed action plan to start process improvement activities. Some improvement activities can begin soon after the completion of the on-site assessment. The implementation of certain improvements is an important motivation factor for all members of the organisation.

During the assessment, it is recommended to collect both quantitative and qualitative data (i.e. indicators) which will be used later to measure the progression realised. One could obtain data on non respected budgets and schedules, or measure the degree of satisfaction of the customers regarding product quality level. Since senior management will have made investments, it is very appropriate to be able to demonstrate that these investments have been profitable.

Lesson 5 : Train all Users of the Processes Methods and Tools

Once the processes defined, it is essential to train all users. Otherwise, all related documents will end up getting dusty on shelves. It is illusory to think that developers will study, by themselves, new processes in addition to their work load. Training sessions also serve as a message that the organisation is going ahead and will require that its developers use these practices. During the training sessions, it is necessary to indicate that, however everybody's good will, errors are bound to happen while using new practices. This will help reducing developers' level of anxiety in their using these new practices. It would be a good thing that a resource-person be available to help developers when the latter face obstacles while implementing new practices.

Lesson 6 : Manage the Human Dimension of the Process Improvement Effort

The author also wishes to make the reader aware of the importance of the human dimension in a process improvement program. The people responsible for these changes are often extremely talented software engineering practitioners, however not too well equipped in change management skills. The reason for this is simple. During their training, they focused on the technical dimension and not on the human aspect. However, the major difficulty in the whole improvement program is precisely the human dimension. Also while preparing the technical part of the action plan, the change management elements have to be planned (Laporte 1994). This implies, among other things, a knowledge of (1) the organisation's history with regards to any similar efforts, successful or not, made formerly; (2) the company's culture; (3) the motivation factors; (4) the degree of emergency perceived and communicated by (a) the management, (b) the organisation's vision, and (c) the management's real support. The author is convinced that the success or the failure of an improvement program has more to do with managing the human aspect than managing the technical aspect.

Lesson 7 : Process Improvement Requires Additional "People Skills"

In an organisation that truly wants to make substantial gain in productivity and quality, a major cultural shift will have to be managed. Such a cultural shift requires a special set of "people" skills. The profile of the ideal software process facilitator is someone with a major in social work and a minor in software engineering. The implementation of processes implies that both management and employees will have to change their behaviours. With the implementation of processes, management will need to change from a "command and control" mode to a more participative mode. As an example, if the organisation truly wants to improve its processes, a prime source of ideas should come from those who are working, on a daily basis, with the processes, i.e. the employees. This implies that management will need to encourage and listen to new ideas. This also implies that the decision making process may have to change from the autocratic style, e.g. " do what you are told" to a participative style, e.g. "let us talk about this idea". Such a change requires support and coaching from someone outside the functional authority of the manager who has to change its behaviour. Similarly, employees' behaviour should change from being the technical "heroes" that can solve any bug, from being passive and unheard in management issues to work in teams and generate and listen to others' ideas to make improvement.

Also, the first few months of the introduction of a new process, a new practice or a new tool, both management and employees must acknowledge that mistakes will be made. Unless a clear signal has been sent by management and a "safety net" has been deployed to recognise this situation, employees will "hide" their mistakes. The result is that not only the organisation will not learn from them but other employees will make the same mistakes again. As an example, the main objective of the inspection process is to detect and correct errors as soon as possible in the software process. Management has to accept that in order to increase the errors

detection rate, results from individual inspections will not be made public, only composite results from many inspections (e.g. at least ten inspections) will be made public. When this rule is accepted by management, employees will feel safe to identify mistakes in front of their peers instead of hiding them. The added benefit to correcting errors early in the process is that those who participated to an inspection will learn how to avoid these errors in their own work.

Facilitating such a change in behaviours requires skills that are not taught in technical courses. It is highly recommended that the people responsible for facilitating change be given appropriate training. The author recommends a course given by the SEI, the title of which is "Managing Technological Change". For lack of such a course, the author recommends to read two books that may facilitate the management of change: the first one (Block 1981) gives advises to anybody acting as internal consultant; the second one (Bridges 1991) gives the steps to be followed for writing and implementing a change management plan.

8 Conclusion

The Software Engineering Institute's Capability Maturity Model as well as Trillium and SPICE models have been used successfully by some organisations in Québec to conduct assessment and to put in place process improvement programs. As more organisations perform similar activities, we should be in position to verify if these activities will have an impact on software productivity and on organisations' profitability. Finally, let's remember that any improvement process includes a human dimension which, at times, has a bigger impact than the technological dimension, should it be neglected during the improvement phase.

Acknowledgements

The author wishes to thank all ASEC's software engineering specialists, as well as the organisations quoted as examples for their remarks and suggestions.

Electronic References

For more information, please get in touch with the following electronic addresses:

Montréal SPIN - http://www.crim.ca/.cgla/spin/spin-f.html
Applied Software Engineering Centre - http://www.crim.ca/cgla/
IREQ - http://www.ireq.ca/degel
Trillium model - http://ricis.cl.uh.edu/Trillium/

Software Engineering Management Research Laboratory -
http://saturne.info.uqam.ca/Labo_Recherche/Irgl.html

References

BATE, R., et al., 1995, *A Systems Engineering Capability Maturity Model,* Version 1.1, Software Engineering Institute, Technical Report CMU/SEI-95-01, November.

BELANGER, R., 1996, *System and Software Process Improvement (Bombardier Mass Transit Division),* in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

BLOCK, P., 1981, *Flawless Consulting,* Pfeiffer & Company.

BRIDGES, W., 1991, *Managing Transitions,* Addison Wesley.

COALLIER, F., 1995, *Trillium: A Customer-Oriented Assessment Method for Software System Development Capability,* in Metrics in Software Evolution, GMD-Forschungszentrum Informationstechnik GmbH, Report No. 254, Monika Mullerburg, Alain Abran, Publishers. ISBN 3-486-23589-3, published by R. Oldenbourg Verlag GmbH, Munich.

DEPARTMENT OF DEFENSE, 1987, *Report of the Defense Science Board Task Force on Military Software,* Office of the Under Secretary of Defense for Acquisition, Washington, D.C., September.

FOWLER, P. et al., 1990, *Software Engineering Process Group Guide,* Software Engineering Institute, Technical Report CMU/SEI-90-TR-24, September.

HUMPHREY, W.S. et al., 1987, *A Method for Increasing the Software Engineering Capability of Contractors.* Software Engineering Institute Technical Report CMU/SEI-87-TR-23, September.

HUMPHREY, W.S. et al., *State of Software Engineering Practice*. Software Engineering Institute, Technical Report CMU/SEI-89-TR-1.

HUMPHREY, W., 1994, *A discipline of Software Engineering,* Addison Wesley.

HUMPHREY,W.S., 1996, *Using a Defined and Measured Personal Software Process*, IEEE Software, May.

IEEE Computer Society / Technical Committee on Software Engineering (TCSE). *Software Engineering Technical Council Newsletter*, September 1994, Volume 13, No. 1, p. 4.

LAFLEUR-TIGHE, A., 1996, *Formalisation des processus dans un contexte R & D*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

LAMBERT, G., 1992, *"Bénéfices de l'évaluation du Software Engineering Institute"* (Benefits of the SEI's assessment). Minutes of Symposium 92 GERICO, Montreal, May.

LAPORTE, C.Y., 1993, *Process Maturity Models: Canadian Experience*, Metrics in Software Evolution, A German-Quebec Workshop, Sankt-Augustin, Germany, October 18-20.

LAPORTE, C.Y., 1994, *Process Improvement and the Management of Change",* Proceedings: 4th. IEEE Computer Society Workshop on Software Engineering Technology Transfer, Dallas, April 28-29.

LAPORTE, C.Y., 1995, *Software Process Engineering Activities in Québec*, in Metrics in Software Evolution, GMD-Forschungszentrum Informationstechnik GmbH, Report No. 254, Monika Mullerburg, Alain Abran, Publishers. ISBN 3-486-23589-3, published by R. Oldenbourg Verlag GmbH, Munich 1995.

LAPORTE, C.Y., 1996a, *Maturation du génie logiciel au Québec: où en sommes-nous?*, L'expertise informatique, Vol. 2, no. 1.

LAPORTE, C.Y. et al., 1996b, *Software and Systems Engineering Process Improvement at Oerlikon Aerospace*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

MAYRAND, J., 1996, *Risk Assessment for Investment Decisions in Software Venture Capital*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

MARCHOK, J., 1993, *Tapping into the Software Process Improvement Network,* Bridge, Software Engineering Institute, Carnegie Mellon University.

MODAFFERI, C., *Ericsson's Total Business Improvement Program*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

PAULK, M. et al., *Key Practices of the Capability Maturity Model*. Version 1.1, Software Engineering Institute Technical Report CMU/SEI-93-TR-25, February.

PAULK, M.C. et al., 1994a, *ISO Seeks to Harmonize Numerous Global Efforts in Software Process Management.* IEEE Computer, April, p. 68-70.

PAULK, M.C., 1994b, *A Comparison of ISO 9001 and the Capability Maturity Model for Software*, Software Engineering Institute, Technical Report CMU/SEI-94-TR-12.

PAULK, M.C., 1995, *How ISO 9001 Compares with the CMM,* IEEE Software, January.

POULIN, L., 1996, *Software: Process Risks Identification, Mapping and Evaluation - The S:PRIME Approach*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

SAYEGH, S., *Canadian Marconi Company's Software Process Improvement Strategy for a Multi-Product Multi-Site Environment*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

SHOSTAK, B., *Adapting Elements of the PSP in an Industrial Environment*, in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

ZITOUNI, M. ET AL., 1996, *Un modèle pour évaluer et améliorer la qualité du processus de maintenance des logiciels,* in Proceedings Process Improvement Symposium: Toward an International Vision, Centre de Recherche Informatique de Montréal (CRIM), ISBN 2-921316-63-3.

ZUBROW, D. et al., 1993, *An Analysis of SEI Software Process Assessment Results.* Minutes from the Software Engineering Symposium, SEI, Carnegie Mellon University, Pittsburgh.

The Hungarian Quality Scene - Potential for Co-operation

Miklós Biró, Éva Feuer, Tibor Remzső

Computer and Automation Institute
Hungarian Academy of Sciences
Budapest, Lágymányosi u. 11. H-1111 Hungary
Tel +361 269 8270, Fax +361 269 8269
e-mail: miklos.biro@sztaki.hu

Introduction

The objective of this paper is to encourage the mutually beneficial cooperation between software and systems development organizations of Hungary and other countries. The Strengths, Weaknesses, Opportunities and Threats (SWOT) are analysed from the perspective of the possible exploitation of achievable financial, operating, marketing, and production leverages with Hungarian software firms. The benefits for cooperating partners are the following for example: highly educated workforce, cultural proximity, relatively low cost. Nevertheless, cooperation is difficult to initiate because of perceived threats arising from the former neglect of the development of a quality culture. This paper highlights the emerging trends in the Hungarian quality scene, and introduces a new opportunity for establishing mutually beneficial cooperation projects. The idea is to make use of ISCN as a coordinating agent with the role of assessing and improving the quality systems of potential Hungarian partners, reducing in this way the risks of other partners to cooperate with them.
In the following we give an overview of the Hungarian information technology market in general and the software market in particular in order to introducing the environment in which Hungarian software firms are operating. A short SWOT analysis of Hungary in the software development area is presented next, followed by the results of a survey of the quality awareness of Hungarian software producing firms.

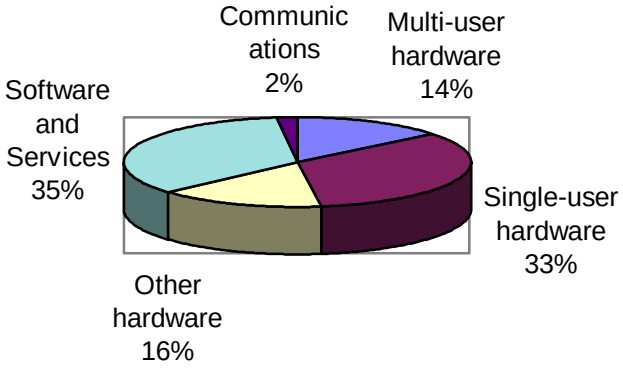
The Information Technology Market

Size of the Hungarian Information Technology Market (1993)

US$ 610.4M

Relative Size of the Submarkets

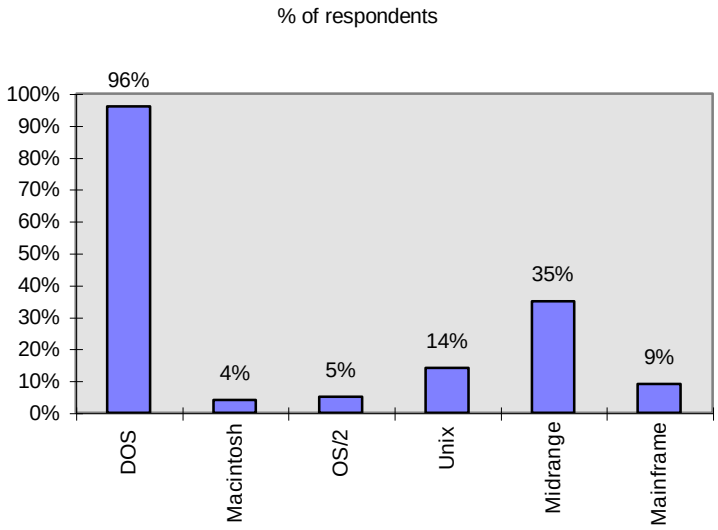
Figure 1.



Data source: IDC Ltd.

Hardware Installations (1994)

Figure 2.

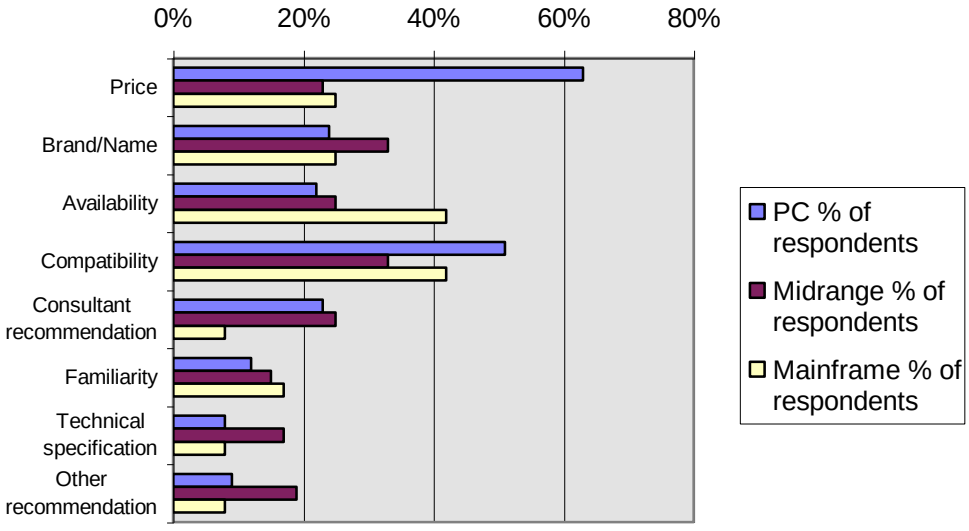


Data source: Deloitte & Touche, IDOM

The dominance of DOS based PC's is primarily due to their relatively low price, even though their original market penetration was determined by the past CoCom restrictions on the transfer of high technology.

Driving Forces of the Market

Figure 3.



Data source: Deloitte & Touche, IDOM

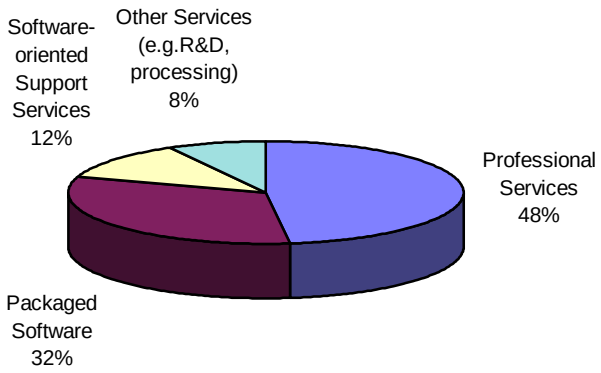
The Software Market

Estimated Revenues of the Hungarian Software Industry from the *Software and Services Market* (1993)

US$ 112.2M

Estimated Relative Revenues of the Hungarian Software Industry from the *Submarkets of the Software and Services Market* (1993)

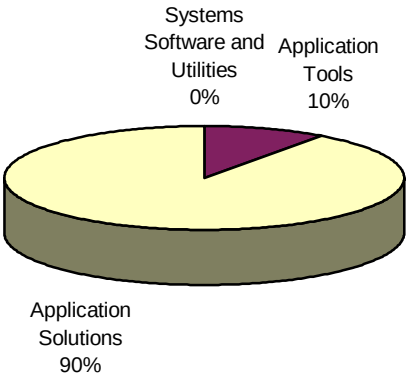
Figure 4.



Data source: IDC Ltd.

Estimated Relative Revenues of the Hungarian Software Industry from the *Submarkets of the Packaged Software Market* (1993)

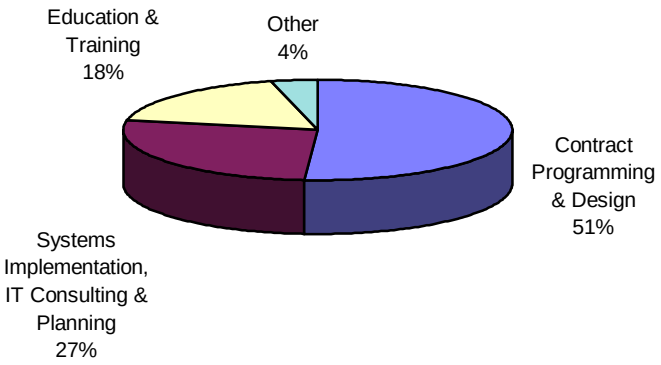
Figure 5.



Data source: IDC Ltd.

Estimated Relative Revenues of the Hungarian Software Industry from the *Submarkets of the Professional Services Market* (1993)

Figure 6.



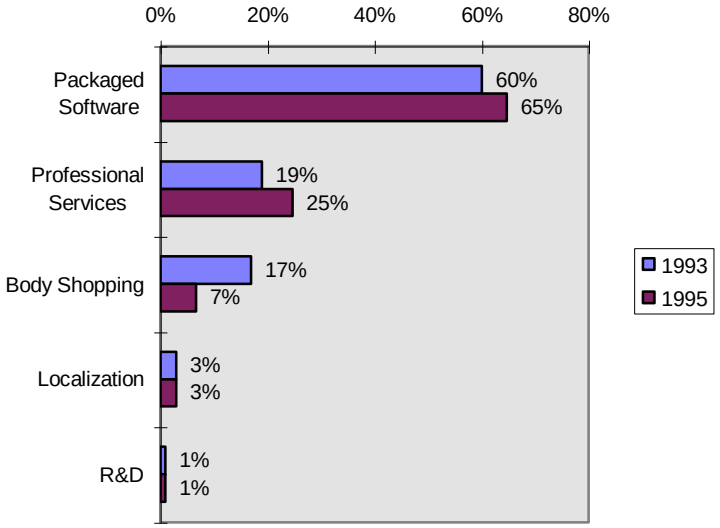
Data source: IDC Ltd.

Estimated *Export* Revenues of the Hungarian Software Industry from the *Software and Services Market* (1993, forecast for 1995)

1993: US$ 25.5M 1995: US$ 32.9M

Dynamics of *Export* Revenues of the Hungarian Software Industry from the *Submarkets of the Software and Services Market* (1993, forecast for 1995)

Figure 7.



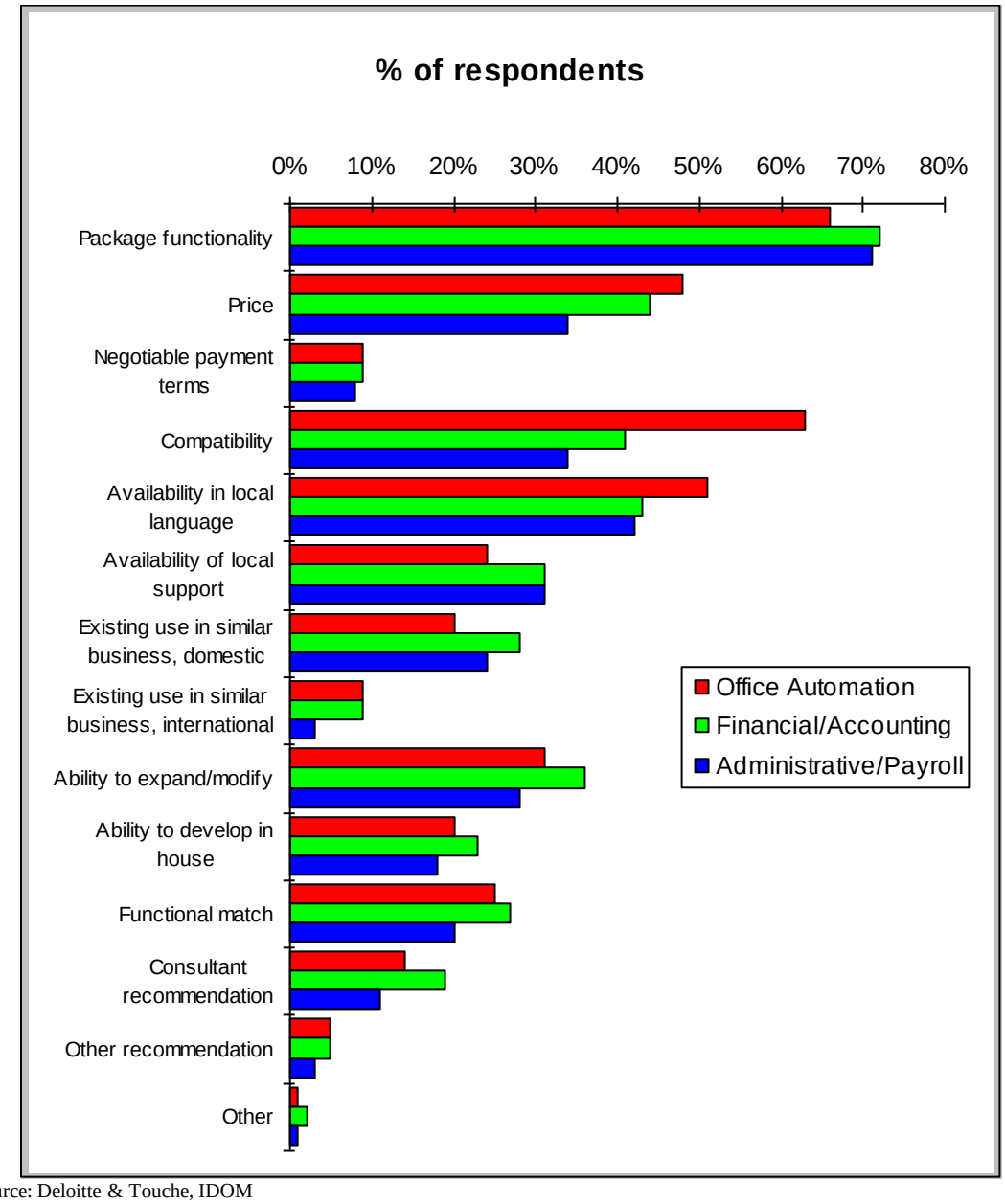
Data source: IDC Ltd.

The growth of packaged software exports is due to the emergence of new technologies requiring relatively low investment, like CD ROM applications.

The growth of professional services is a natural phenomenon accompanying general recovery and the accumulation of professional experiences.
The former magnitude of body shopping was a consequence of one of the weaknesses of Hungarian industry: the relative lack of local managerial skills and experiences. The difficulty of finding enough managers to handle large projects locally, made body shopping the only mutually profitable possibility for foreign firms to exploit the highly educated but low cost workforce. Attempts to resolve this problem by temporarily involving foreign experts often lead to strong dissatisfaction because of recommendations ignoring local conditions. A successful and also mutually profitable solution is a joint venture where local technical knowledge is combined with foreign managerial experience on a long term basis. An alternative or complementary path to success is large scale investment into management related training including quality management.

Driving Forces of the Market

Figure 8.



Data source: Deloitte & Touche, IDOM

SWOT (Strengths, Weaknesses, Opportunities and Threats) Analysis from the Perspectives of the Four Possible Levers of a Firm

Levers are means used by a firm to multiply its resources. It is fundamentally the use of levers which can be accounted for the differences in profitability among firms. The four possible levers of a firm are the financial lever, the operating lever, the marketing lever, and the production lever.

Hungary, as one of the emerging economies in Central Europe, has a number of general strengths including a highly educated workforce able to assimilate new skills rapidly, and able to produce high quality goods at relatively low cost for export. For the same reasons R&D capacity is high as well. Large projects mean new opportunities for both foreign and domestic ventures. These projects are becoming urgent because of the limited possibilities of the earlier economic system.

Operating leverage is the relative change in profit induced by a relative change in volume. Because of the low operating costs, the Hungarian software industry has a high operating leverage, by consequent it can generate more profit than its less leveraged competitors as soon as its volume reaches a given level.

A weakness, already introduced in the previous section, is the relative lack of local managerial skills and experiences. This problem has impact on both the production and marketing leverages.

Production leverage is the rate of growth of profits resulting from cost declines. Production leverage can only be achieved if management is able to properly organize production. Quality management is an important part of this organization.

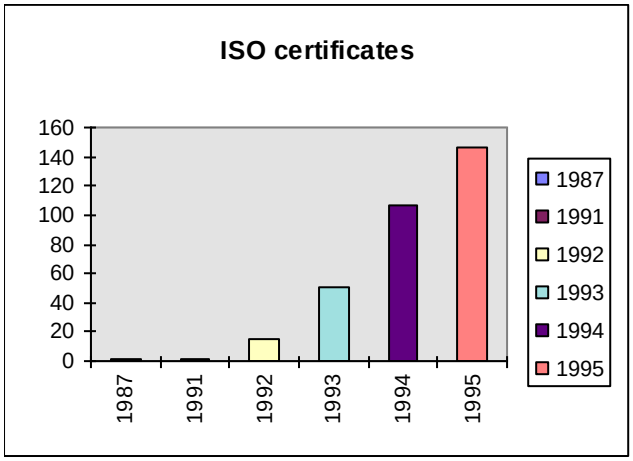
The two main ingredients of marketing leverage are higher prices and innovative distribution. The achievement of any of these goals requires advanced market management skills.

As far as production and marketing leverages are concerned, Hungary is making efforts in training managers to the necessary skills that were unheard of in the former economic system. The possibility of making use of financial leverage, that is having and exploiting debt capacity, depends on the advent of general economic recovery and lower inflation, which is a rather long-term process.

Quality Awareness in Hungary

The general Hungarian Quality Scene is best characterized by the increasing number of ISO 9000 certifications depicted on the figure below.

Figure 9.



Up to now however, there are very few software development organizations which have achieved ISO 9000 certification. We have information about some of them being in the process of developing their quality systems.

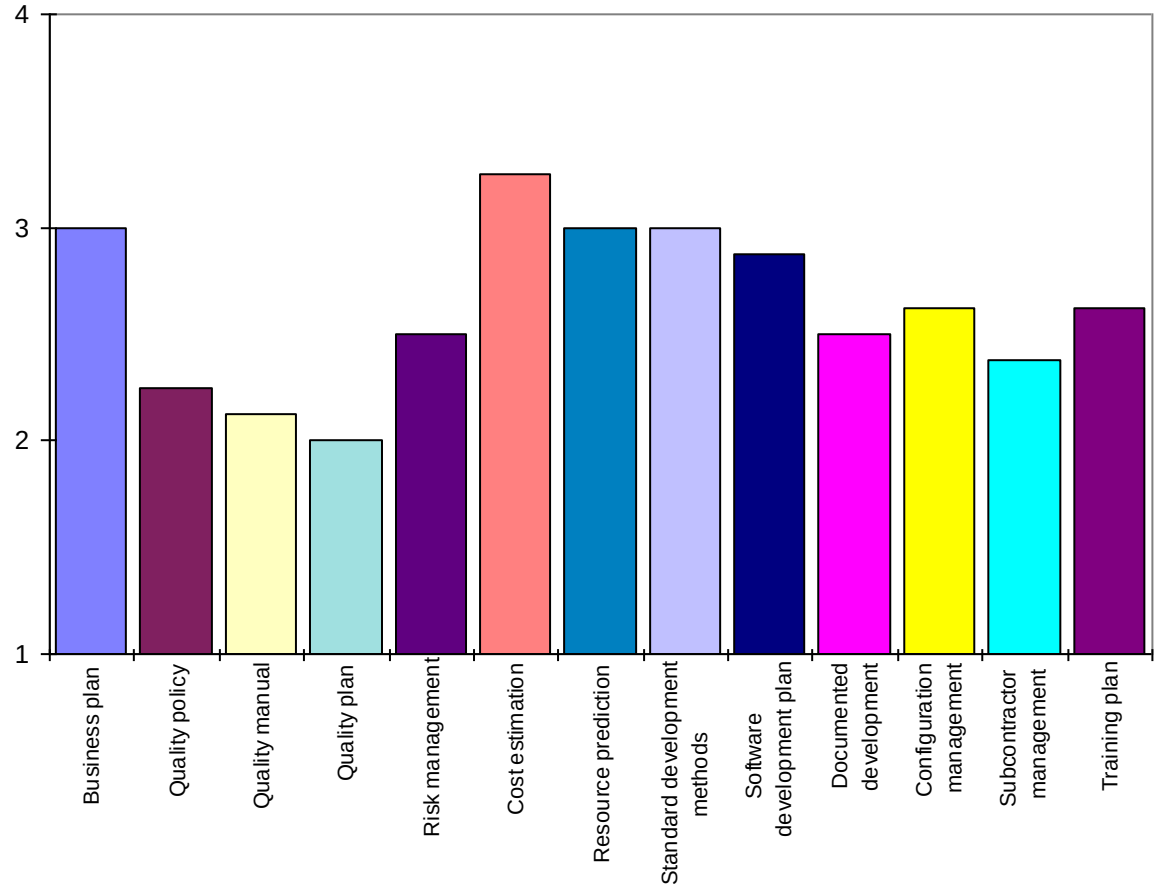
As far as the capability maturity of software development firms is concerned, we assessed some software companies with the help of the BOOTSTRAP software process assessment methodology which is the market leader in Europe. According to our assessments, the maturity levels of assessed software producing units were between 1.25 and 2.75.

In order to getting a broader picture of the quality awareness of the Hungarian software industry, we created a short questionnaire. Companies were asked to reply voluntarily and anonymously. Completed forms were forwarded to us by various means (Internet, fax, mail). The replies received were statistically analyzed.

88 percent of respondents knew about ISO 9000 standards, 38 percent knew the BOOTSTRAP methodology. A few have heard about CMM, SPICE and TickIT methodologies and standards, other methodologies were not well known. The demand or requirement for formal certification has not become obvious yet. The majority of respondents (88 %) does not or rarely requires formal certification to ISO 9000 from their subcontractors. Usually they are not required to have formal certification as a subcontractor, either. At the same time, the majority of respondents feel the need for the formal certification of their quality management system. Some of them are planning a certification or are currently undergoing one. The initiations of quality management are present almost everywhere.

The second half of the questionnaire is directed towards the specific areas of quality management. Questions are asked about the level at which processes of a specific area are accomplished or the existence and level of detail of certain documents. Answers could be chosen from a range of four levels. Results are of course not precise enough to conclude at some general maturity level, but are satisfactory to make comparisons between awareness in the various quality areas. The following chart shows the results of this part of the questionnaire. Level 1 means, the process or task is not performed or the documentation does not exist, level 4 means that the process is fully performed and the documentation is complete.

Figure 10.



The coordinating role of ISCN

ISCN is an Irish firm whose key asset is a pool of experts who represent a wide range of approaches and methodologies allowing a synergetic combination of the skills most suitable to the specific requirements of its customers. ISCN is by consequent well positioned to vitalize a coordination model where mutually beneficial and reduced risk cooperations could be established with Hungarian and other Central and Eastern European software developing firms.

The coordination model is based on the following process:

- ISCN cooperates with the Computer and Automation Institute of the Hungarian Academy of Sciences (MTA SZTAKI) in establishing an expert skill database across Central and Eastern European countries using a Procedure Quality Manual for certifying experts.
- MTA SZTAKI uses the expert pool to evaluate the capability of Hungarian and other Central and Eastern European firms and to register those who have a capability maturity level above 2.5. This means a satisfactory level of cooperation risk for partners who may also obtain more detailed maturity profiles if necessary and agreed by all parties.
- ISCN and MTA SZTAKI promote the outsourcing cooperation and support the establishment of the corresponding contracts.

The above model makes it possible to exploit the opportunity of higher production leverage for Hungarian and other Central and Eastern European firms and of higher operating leverage for partner firms.

Conclusion

The maturity level for quality software development of Hungarian firms is changing rapidly. ISCN is ready to play the role of coordinating agent in establishing mutually beneficial and reduced risk cooperations with Hungarian software developing firms.

Even though most of the analysis concerned Hungary only, it is clear that similar processes are going on in most Central and Eastern European countries. In order to satisfying the specific needs of its customers, ISCN is also ready to mobilize its relationships in these other countries with high software development potential.

References

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

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

[3] Biró,M.; Remzsô,T.; Sz.Turchányi,P. Assessment and improvement of the quality of software processes from a decision maker's perspective. In: Proceedings of the VI. National Congress of the John von Neumann Society for Computer Sciences (ed. by I.Tóth). (John von Neumann Society for Computer Sciences, Siófok, Hungary, 1995) Vol.2. pp.671-678. *(in Hungarian)*

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

[5] Biró,M.; Sz.Turchányi,P. Software Process Assessment and Improvement from a Decision Making Perspective. ERCIM News (European Research Consortium for Informatics and Mathematics) No.23 (1995) pp.11-12.

[6] Biró,M. IT Market and Software Industry in Hungary. Keynote paper at the European Observatory on Software Engineering CQS '95 Conference (Rome, 1995).

[7] Biró,M.; Feuer,É.; Remzsô,T. The Bootstrap Institute: International Organization for Software Quality Assurance in the European Union. In: Proceedings of the IV. Quality Week International Conference (ed. by M.Pákh, A.Róth, T.Turos, Á.Hargitai). (Hungarian Quality Society, 1996) pp.133-136. *(in Hungarian)*

[8] Better Software Practice for Business Benefit: Principles and Experience. (book to appear at Wiley).

The Siemens Improvement Method - From Assessment to Improvement Strategies

Dr. Tilo Messer[1]
Siemens AG
Corporate Research and Development
Application Center Software
e-mail: Tilo.Messer@zfe.siemens.de

Abstract
The Siemens Process Assessment has been presented in detail at ISCN'94 [6]. It is based on the Capability Maturity Model (CMM) of the Software Engineering Institute (SEI) [3]. The focus of this paper is on the connection between results of the asseessment and the succeeding improvement program.
The following topics will be discussed:

- The structure of results which come out of an assessment includes an action portfolio which is an important link between assessment and improvement.
- Six basic aspects for successful improvement programs are explained. The first one is to manage an improvement program as a real project.
- Finally experiences with improvement projects will be discussed. Seven improvement rules are listed to successfully bridge the gap between assessment and improvement as well as to role out improvement measures within an organisation.

Introduction

The Application Center Software is part of the Siemens Corporate Research and Development located in Munich. Our main task is the enhancement of the quality and productivity of SW-, Systems and Construction Engineering within Siemens divisions and Siemens Operating Companies world wide. The focus is on the underlying processes, the quality management, and the project's management (see figure 1).
Main products of the Application Center Software are the Siemens Process Assessment and the support and coaching of process improvement programs within Siemens divisions.
The Siemens Software Process Assessment which has been developed by ACS has been presented in detail at the ISCN'94 [6]. It is based on the SEI Capability Maturity Model (CMM) and the BOOTSTRAP results. Furthermore ISO 9000- and special company-related aspects are considered [1,2,3,4,5]. By the Siemens Process Assessments an overall maturity level is computed for reasons of international compareability and for setting global goals. Main effort is spent on

- elaborating detailed profiles and descriptions of strengths and weaknesses throughout 23 key process areas,
- deriving recommendations for improvement actions,
- clustering, priorizing, classifying, scheduling the recommendations to generate a stable basis for an improvement program.

[1] Postal address of the author: Siemens AG, ZFE T ACS, Otto-Hahn-Ring 6, D-81739 Munich, Germany phone: ++ 49 89 636 - 46470, fax: ++ 49 89 636 -44424.

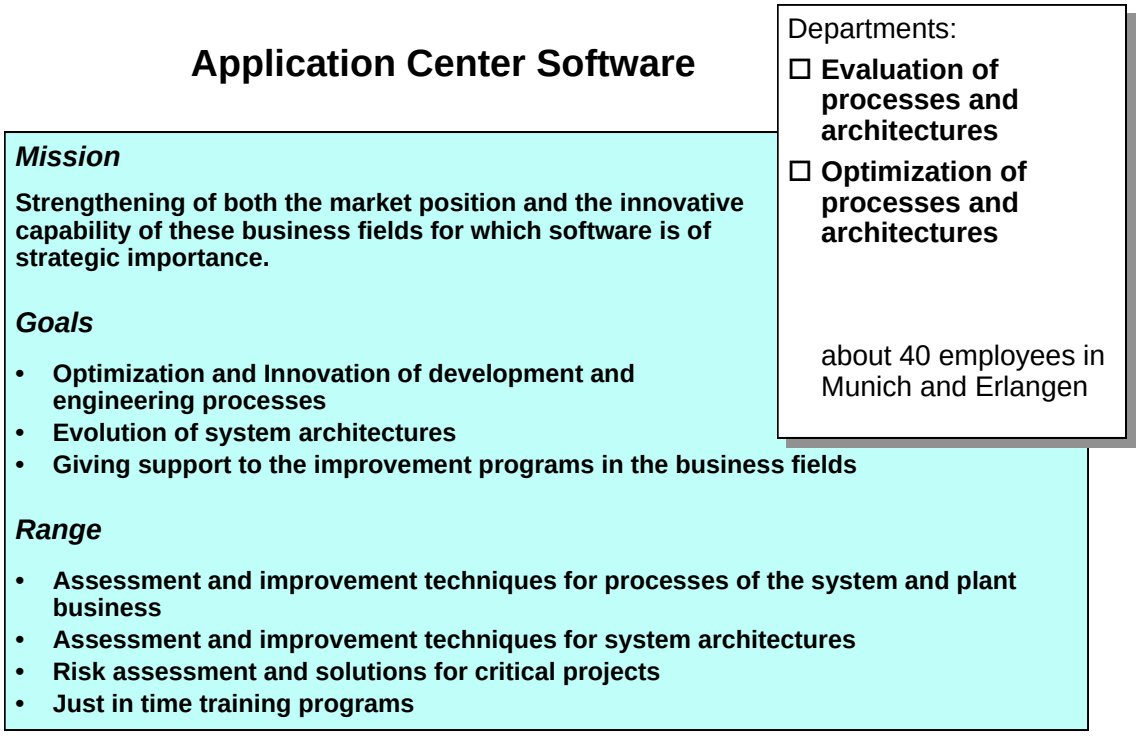


Figure 1: Profile of Application Center Software

The focus of this paper is on the support of the Siemens divisions when they start and perform their improvement programs and on the experiences with the coaching activities. Based on the structured recommendations from the assessment the Siemens division starts an improvement program which is coached by improvement managers of the Application Center Software. Steps for starting an successful improvement program are:

- to identify the strengths and weaknesses of the Siemens division,
- to structure the recommendations following identified potentials,
- to build manageable clusters of activities,
- to define a measurable, realistic and motivating goal of the improvement program,
- to provide strong support and visible sponsorship by the (senior) management,
- to start the improvement program by a kick off meeting.

Action Focused Assessment Approach

The Siemens Process Assessment approach is put under the umbrella of the Siemens top-initiative (‘time optimized processes’) [8]. Goal of this initiative is to improve the competitiveness of all divisions of the company for reaching top positions on the global market. For details of the Siemens Assessment Approach see [8].

The main focus of the Siemens process assessments is on process improvement. The process improvement measures are closely connected to explicitly declared needs, requirements and business goals of the assessed organisation. Therefore 23 key process areas (see figure 2) covering the main aspects regarding organisation, process, project management and engineering have been defined in order to be able to give a complete, detailed and structured result to the assessed organisation. The key process areas define the scope of the improvement measures. In the following, we will explain the most important items of final result reports given to the assessed organisation as well as to the assessed projects. The assessment results of the

projects' processes (esp. findings and recommendations) are integrated into the assessment result of the organisation by making them anonymous.[2]

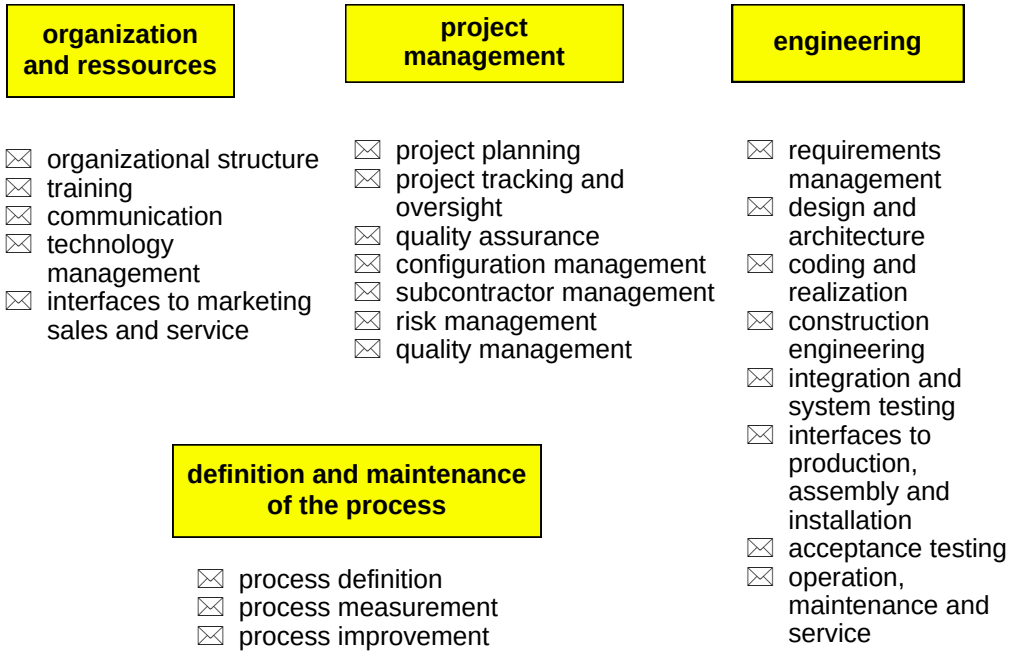


Figure 2: 23 key process areas of the Siemens Process Assessment approach

For each key process area a detailed strength and weakness profile (see figure 3) shows the maturity and the improvement potential. Specific findings and recommendations for each key process area are tailored to the requirements of the organisation and listed in the final report.
The measures (usually about 50) are listed in a measures catalogue. They are connected with a priority (A, B, C), estimations of the effort needed to realize the recommended action and a time frame on the realization / implementation of the recommendations. The priorization and the estimations are done by the assessment team. The assessment team usually consists of two assessors of ACS and additionally up to two assessors by the assessed organization.[3]
Additionally special criteria (e.g. action is relevant for ISO 9001 certification; action is relevant for reducing time-to-market) depending on the assessed organisation and on its goals help to define an organisation specific dimension for defining the improvement program (see figure 3).

[2] Making the results of the projects' processes evaluation anonymous avoids to compare the projects on key process area level. A direct comparison is not beneficial to a motivating and constructive atmosphere between the colleagues in the projects which are in the same boat when they are going to improve their processes.

[3] Each assessor have to successfully pass a CMM training and a training on the Siemens Process Assessment. Both trainings are led and done by ACS.

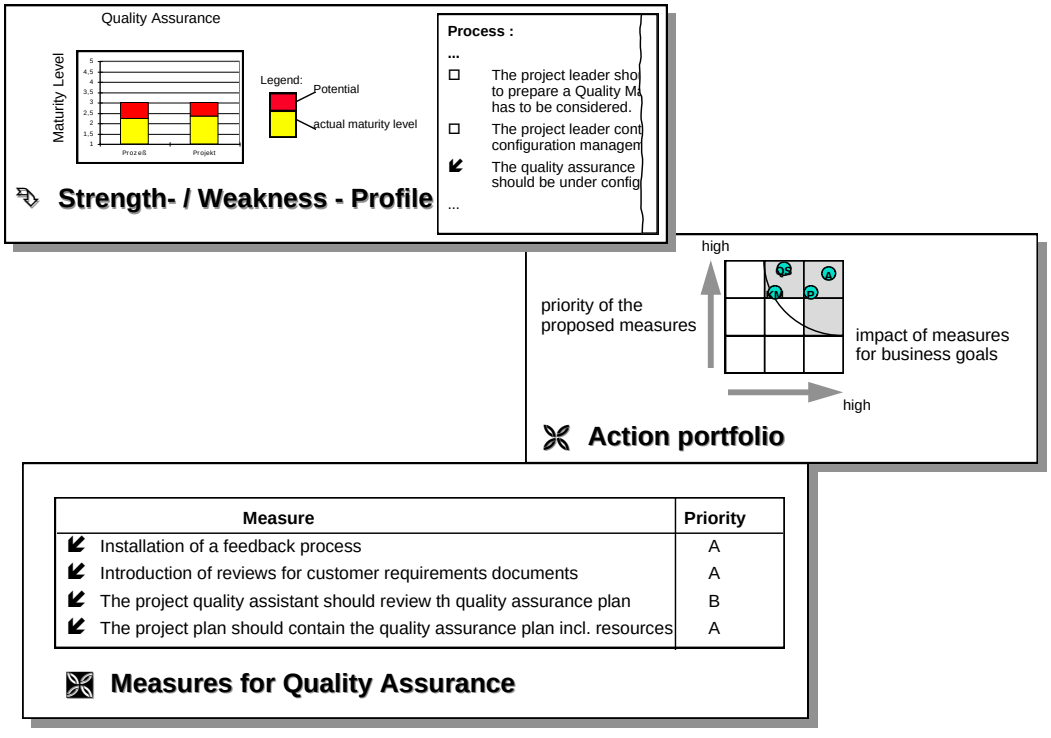


Figure 3: 3 steps during the assessment towards an improvement program (e.g. key process area 'quality assurance')

Naturally, overall maturity levels for the projects' processes as well as for the organisation's process are given. The overall maturity level is in general - and especially for the managers - very helpful but not a key result when the organisation wants to improve its development process. The improvement is not based on the overall maturity level but on the recommendations of the assessment team which are derived from the weaknesses and which are in line with the needs and goals of the assessed organisation.

To come to an higher level of abstraction and a condensed view of the measures in the key process areas clusters are defined which cover a number of interrelated measures. We call these clusters action clusters. Usually about 10 action clusters are agreed between the assessment team and the management of the assessed organisation. They play a key role in this phase of the Siemens Process Assessment. Action clusters are mainly built by clustering some key process areas. For instance, action cluster *organisation* include measures belonging to the key process areas *organisational structure*, *communication*, *training*. Action clusters can also be built across the borders of the key process area columns (we call them key process themes). Each recommendation is located in no more than two action clusters. Experience has shown that two action clusters sufficiently consider the impact of a measure on the improvement program.

The action clusters are positioned in a portfolio which expresses the necessity for implementing measures belonging to an action cluster (see figure 3). Two preconditions must be met to build the action portfolio: first, the measures are priorized - one dimension - and second, the managers give their input to the importance and impact for their business when measures of an action cluster will be implemented in the organisation - the other one dimension.

The action portfolio shows the connectivity between the ranking of action clusters by the assessment team and the ranking of action clusters by its management considering the goals of the business, the vision and the strategy of the assessed organisation. The action portfolio is used as a filter to identify actions with highest benefit and impact on succeeding improvement projects (see figure 4) .

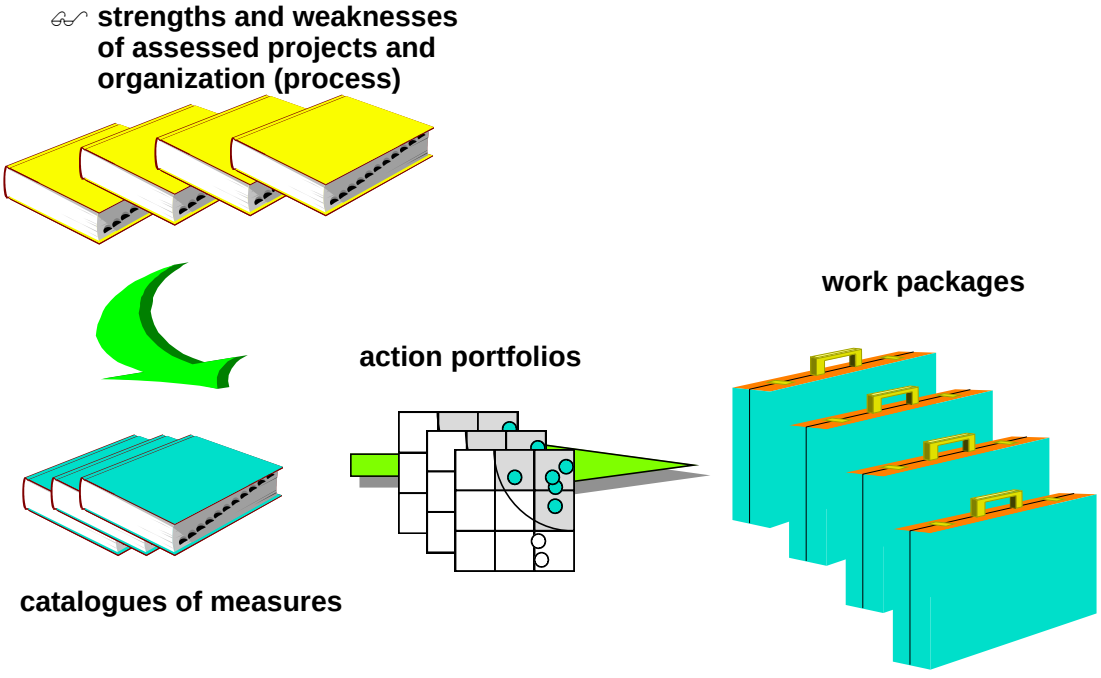


Figure 4: The action portfolios are a filter for the definition of the work packages.

Improvement program - improvement project

The work packages are a very sound basis for a succeeding improvement program. They include actions which have the highest impact on the process improvement. After having the work packages defined the improvement program has to be managed as a project to be successful. This means that

- a project leader has to be appointed,
- adequate resources, budget, and time have to be available for the improvement project,
- mentors for the work packages have to be nominated (mentors are senior managers with a very visible and strong support for the improvement project, e.g. open door policy, capability to make and support decisions in accordance to the goals of the improvement project),
- a measurable goal for the whole project is defined and is agreed by the project team (process improvement team) as well as by the project board (the process improvement board controls the improvement project and supports decisions that influences the process in the organisation; the mentors mentiones above are members of the board),
- planning of the project must be based on the agreed goal and on estimations about size of work,
- and finally, it is highly recommended that results are measured.

Let me pick out one of these factors: a very important factor for the success of an improvement project is the visible and strong support by the senior management. Senior management should follow an open door policy, should provide not only mental support but strong financial and a being-able-to-decide support. Experiences from our improvement projects show that this point is often underestimated and as a consequence to this high effort is spent with only minimal result and/or frustrated team members.

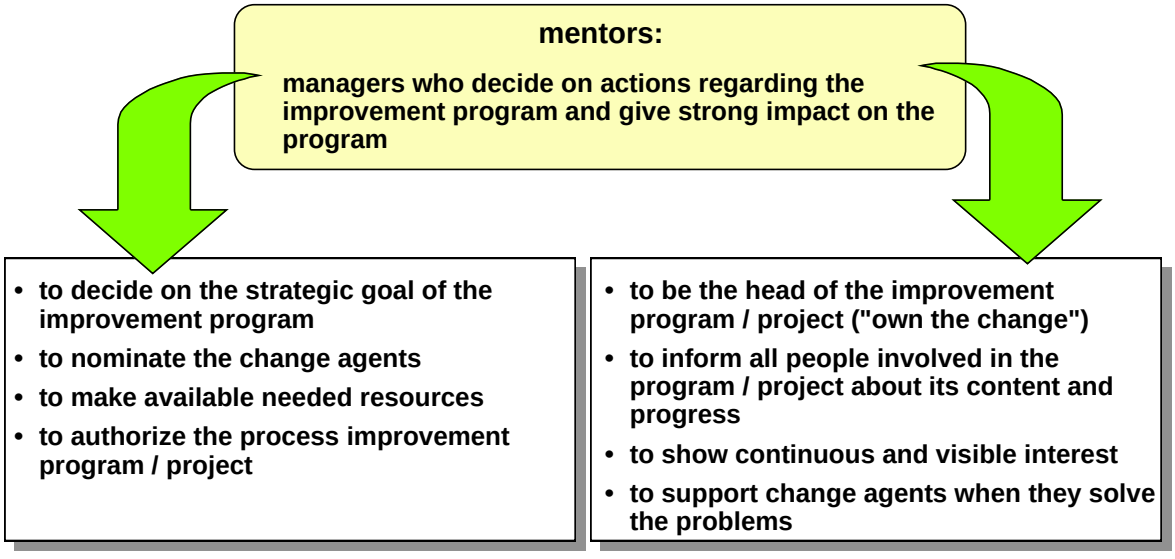


Figure 5: Mentors have an important role during the improvement project

This sounds comprehensible and very easy to implement. But in daily work, the view of the managers is directed to operational goals. Improving the process is not a short term goal for the company. Thus many conflicts appear when they themselves nominated to a mentor. Let me list some examples:

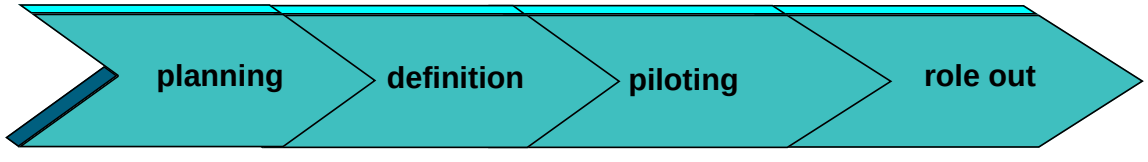
- cost reduction vs. increasing quality costs (quality is not for free),
- customer visit vs. meeting with the improvement board,
- reducing number of organizational instructions vs. fixing documented procedures,
- following planned project's schedules vs. training of people.

It is obvious that not all conflicts can be solved following the improvement project. But the managers should be aware of their arguments why they decide for or against the improvement project's view.

Structure of improvement projects

Improvement projects usually follow 4 phases mentioned below:

- planning of the project,
- definition of goals and pilot applications,
- piloting the selected projects and
- roling out established procedures which are introduced in the selected pilot application throughout the whole company.



- to define goals of the improvement project
- to start the improvement project
- to build the process improvement team
- to plan working packages in detail

- to show status quo
- to modify procedures and make them available
- to select pilot projects
- to plan pilot applications

- to train
- to realize and to coach procedures in pilot projects
- to measure results, to give feedback and to optimize procedures
- to prepare role out

- to modify process documentation
- to train
- to realize optimized procedures in departments
- to measure results and to control progress

Figure 6: 4 phases of an improvement project

There are three options for the participation of the Application Center Software (ACS) in business field specific improvement programs at Siemens:

1. Especially large organisations with sufficient qualified resources are able to plan and implement a program - sometimes - without external help. In this case a deadline for reaching substantial improvement goals is fixed and a reassessment takes place after this time. The reassessment is usually asked for by the organisation itself.
2. In other organisations there is often a lack of knowledge about special topics where weaknesses have been identified in the assessment. In this case the ACS has standard concepts available for some topics which can be adopted to organisation specific requirements and can be introduced quite fast (e.g. configuration management, formal inspections, risk management, QA for external deliveries, defect prevention, project planning and tracking, requirements management, metrics). This also includes just-in-time training and coaching activities. For other topics co-operation with technology deparmtens of our central research labs is available.
3. For smaller organisations the availability of resources and expertise for an improvement program can be a problem. In this case the whole program is planned, managed and realized by ACS.

Experiences with Process Improvement Projects

Up to now more than 70 projects' processes have been covered by Siemens Process Assessments. For most of them an improvement program started soon after the assessment has been finished. The following branches (examples) are covered by assessments as well as improvement programs:

- telecommunications (switching, mobiles, devices)
- power supply and power distribution
- automation
- traffic systems
- medical systems
- automotive control systems
- installation systems
- air control systems
- workstation development (operating systems)

The assessments have taken place at company sites in Austria, Brasil, France, Germany, Italy, Switzerland, U.K. and USA.

Regarding the improvement projects which succeed after an assessment we have made the following experiences:

The improvement project should be started immediately after the assessment has been finished.
If the time slot between the assessment and the improvement project is too large the motivation to start decreases dramatically because of daily work is getting continuously more important. The management has to be convinced once again to support strongly the improvement project. We try to start the improvement project by a regular kick off meeting within at least two months after the assessment has been finished.

Detailed results of an assessment are a sound basis for an improvement project.
The number of recommendations elaborated during the assessment are typically quite high (about 50 - 100 measures depending on the number of assessed projects). This seems to be a disadvantage according to the acceptance of the overall assessment result but it comes the other way round. Because of the action clusters which are proposed by the assessment team and which are agreed by the management of the assessed organisation the recommended actions can be presented in a more compact way.
A rough planning of the improvement project during the assessment (typically at the end or after meaures have been settled) also increases the acceptance and willingness to improve and also supports the existing motivation of the employees to start an improvement project in time. This motivation is based on the action focused approach of our process assessment.

Participants of assessments support the improvement project themselves.
The employees who are directly involved in the assessment have the possibility to propose improvement activities and measures regarding their own work environment. These improvement proposals are directly put into the results of the assessment. They clearly see the advantages of improvements and of activities which overcome barriers and make things easier. The more extensive developers are interviewed the higher the acceptance of the recommended improvement activities is.
It is helpful to involve in the process improvement team (project team of the improvement project) colleagues who have been participated in the assessment, champions who have deep technical experience and expertise as well as opinion leaders who are key persons for getting wide support within the organisation.

Success has to be measurable.
The improvement project is successful if the improvement actions which are realized lead to significant increase of efficiency in daily work. This means that every person involved in a pilot project at least must get the feeling (measurable?) that his work can be done more efficiently caused by the improvement actions.

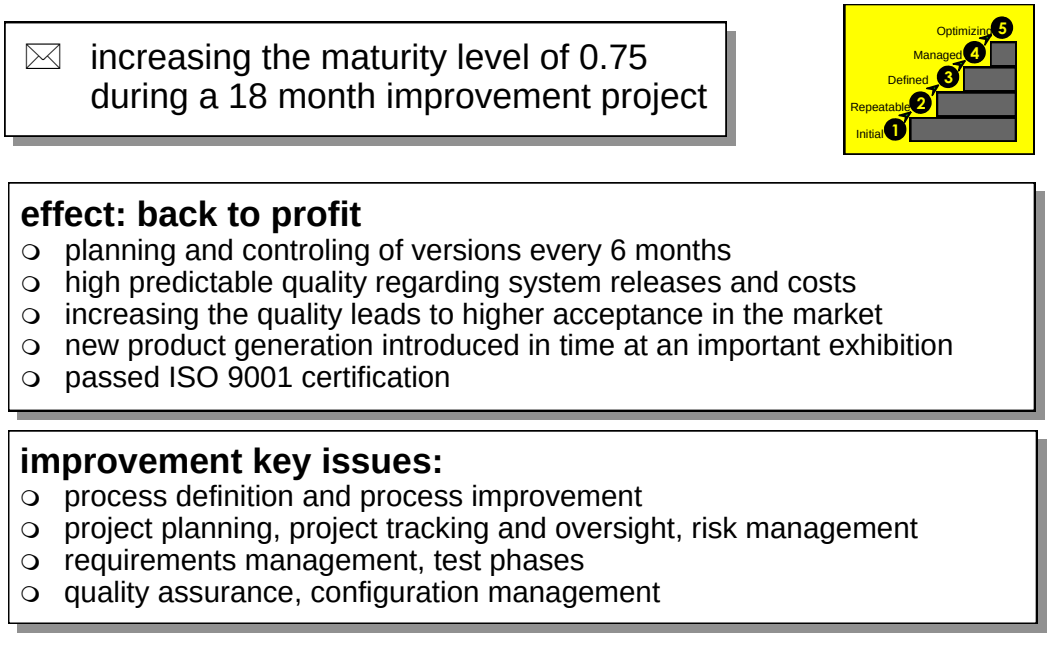


Figure 7: Example of an successful improvement project within Siemens.

The success of the whole improvement project has to be measurable (e.g. ISO 9001 certification passed) (see figure 7). A measurable goal for the improvement project has to be defined (e.g. reaching level 3 within 24 months; measured by an re-assessment - reduction of test cycle time by factor n, reduction of development costs to n DM, reduction of error rates after delivery fewer than 50 dpm (defects per million); the last 3 goals could be measured by introducing a metric system).
But metrics in general is a problem. Often the organisation does not have collected data correctly and completely for a goal leading approach. A clearly defined connection between goals of the company (strategic goals, operational goals) and goals of the improvement project must exist. This is a base for a quantitative tracking of the success of the improvement project.

External coaches help to speed up the project's progress.
The improvement managers of the ACS are considered to be 'external coaches' when they are involved in or when they manage improvement projects in Siemens divisions. It is helpful to have external coaches because they are not integrated in daily work and in problems of the organisation. Additionally they have the possibility and acceptance to gap bridges between established groups of developers in order to come to agreed results and decisions. The continuous support of external coaches can be decisive for the success of the improvement project.

The whole organisation should be kept informed about the status of the improvement project.
It is essential that all people who are directly and indirectly involved in the improvement project are regularly kept informed about the planning, status and the goals reached (success stories) of the improvement project (e.g. by sending out newsletters every two or three months, by organizing open discussion sessions where everyone gets the possibility to ask questions and critizise - with and without managers, to propose new improvement actions). The culture of change must be visible to everyone in the company.

Established improvements in pilot projects can be lost.
Improvements have to be introduced step by step. Pilot projects are suitable candidates for establishing improvement actions in a smaller environment. Coaches have to be aware of spreading effects and experiences of improvement actions across the border of the pilot projects. In other case improvement actions in pilot projects grow and bloom but die after the project has been finished.
It is a main part of the improvement project to plan how to role out all improvement activities throughout the whole organisation.

Conclusion

It seems that our experiences made are only problems, but not more. This is not quite true. Assessments as well as improvement projects bring fun although daily improvement work is hard. We have learned that improvement projects become a success if at least the improvement rules mentioned above are strictly considered and realized.
Quality is not for free, improvement neither. In average the costs of an assessment based improvement project (including assessment, investment, training, project team of the organisation, introduction of measures) are about 3-7% of the development costs of the organisation. The other way round improvement projects lead to an increase of efficiency of about 30% within 2-3 years.
It is difficult to measure cultural change. But process improvement makes cultural change necessary in order to be competitive and successful in changing markets.

I would like to give specials thanks to Thomas Mehner and Axel Völker for fruitful comments and discussions as well as for their reviews of the draft version of this paper.

Literatur

[1] Watts S. Humphrey, Managing the Software Process, SEI, Addison Wesley, 1989
[2] SEI, The Role of Assessment in Software Process Improvement, CMU/SEI-89-TR-3, CMU, 1989
[3] SEI, Capability Maturity Model, CMU/SEI-91-TR-24, CMU, 1991
[4] SEI, SoftwareEngineering Process Group Guide, CMU/SEI-90-TR-24, CMU, 1990
[5] Software Process Unit Assessment Questionnaire, BOOTSTRAP, 2i Industrial Informatics, Freiburg, 1991
[6] Axel Völker, Software Process Assessments at Siemens as a Basis for Process Improvement in Industry, Proceedings of the ISCN'94 Conference, ISCN Ltd., Dublin, Ireland
[7] K.H. Möller / D.J.Paulisch, Sortware-Metriken in der Praxis, Handbuch der Informatik 5.4, Oldenbourg Verlag München, 1993
[8] Thomas Mehner, Axel Völker, Siemens Process Improvement Approach, Proceedings of the ISCN ‘95 Conference, ISCN Ltd, Vienna, Austria

From Business Goals to Improvement Planning: Practical Use of ami® in Industry

Christophe Debou
Alcatel Telecom
Excelsiorlaan 44-46
1930 Zaventem
Belgium

☎: +32/2 718 70 22
Fax: +32/2 718 75 99
💻: Christophe.debou@nsgzm.alcatel.be

Annie Kuntzmann-Combelles
Objectif Technologie
28 Villa Baudran
94742 Arcueil cedex
France

☎: +33/1 49 08 58 00
Fax: +33/1 49 08 95 88
💻: akc@objectif.fr

The Software Process Improvement (SPI) field is loosing credibility along the time. This is due partly to a misuse of the main concepts, partly to a too theoretical talk from the SPI suppliers. The critical phase i.e. to devise an action plan based on the assessment findings and the business strategies is being underdocumented. This article intends to give more insight into this problematic and to describe how the application of a goal-oriented approach like **ami®** can support the action planning phase of an SPI programme.

Introduction

The major Software Process Improvement (SPI) conferences and forums have highlighted the lack of improvement planning strategies. What the SPI community has been documenting so far almost exclusively relates to the assessment process and the model to evaluate against. Several methods (Bootstrap [HMK94], SPICE [KIT96], ...) have been derived from the original SEI work (Capability Maturity Model, assessment process) [CMM93a, CMM93b, Hum89].

But, when you actually coordinate such SPI initiatives, the real issues situate at the end of the assessment: translation of the assessment results into concrete improvement actions namely improvement action planning. And this exercise is much more tricky than trying to cover straigthly not existing and deficient CMM practices. This is what we will try to address in the present article by introducing the **ami®** approach (Application of Metrics in Industry), one possible strategy for structuring improvement program, driven by goals and metrics [DFH95, DKR94]. What has to be avoided is that the SPI group is disconnected from the projects, like the quality assurance group often was.

After detailing the problematic i.e. “to bridge the gap between assessment and improvement actions”, the **ami®** method will be described in the context of SPI. Then two case studies from industry will highlight software process improvement initiatives where the application of **ami®** concepts has positively triggered the action planning phase.

Problematic: to bridge the gap between assessment and improvement actions

The 95' SEPG (Software Engineering Process Group) conference in Boston [SEPG95] and the recent European SEPG conference in Amsterdam confirmed that two third of the SPI initiatives in the US did fail, meaning their impact on the organizational performance was low or unexisting (no statistics do exist in Europe so far due to the much lower number of organizations committing to such program but the author's experience will probably come to the conclusion that 2/3 of the initiatives do not come to the end properly even if they do not fail). The two main reasons were the changes of business situation (basically, organization bought by another, merge between two groups, ...) for one third and the lack of management sustain for another one third. This lack of senior management support is mainly due to a lack of visibility towards potential benefits and actual return on investment in industry but also to an inadequate way of handling such programs which are viewed more as a theoretical exercise than a critical item for the business. Other reasons mentioned for the non-success are:

- Mid management resistance to change; on the one hand, they are asked from their boss to improve productivity and on the other hand, from project to obtain more resources to reduce delays: so "*what the hell with SPI"* . Their career development is more linked to the margin observed on projects than to the SPI effort.
- SEPG not having the right skills - another consequence that the management does not take the initiative so seriously
- SPI not managed as a project (one of the most important project) with clear objectives
- Late action plan: momentum is lost both at management and practitioner level.
- No real links with business objectives.

In fact, to our mind, the latter issue causes most of the trouble: Software Process Improvement is not perceived as a business issue.

Whose fault? There is usually an overemphasis on the assessment process, maturity questionnaire, maturity profile and the Capability Maturity Level (CMM). The ISO 9001 Syndrome (to document a posteriori processes/procedures for sake of certification) starts to apply for the SEI CMM/Maturity level. Concepts are sometimes misused when utilized for "certification purposes". This happens and hurts the whole SPI community. For chance, there exist some organizations respectively senior managers who do perform right, who do understand the concepts (actually the same what was called some years ago total quality management) and do apply it effectively and not "literally". There exist ISO 9001 certified organizations that are living it everyday. When a CMM-based assessment is performed, the purpose is quickly well-understood (organizational structure for improvement already exists), people expresses their satisfaction with their quality system, management has to admit that the investment upfront was high but worthwhile (they measured it). There also exist ISO 9001 or

even TickIT certified organizations that perform crash actions just before the certification to clean up their quality system. A software process assessment reveals the "cheating" immediately while interviewing practitioners: just an overhead activity, shelfware procedures and instructions, ... And one of their main concerns relates to the commitment of management for an SPI initiative: "*Yet another initiative*".

To summarize, it's time to be pragmatic. Some years ago, Vic Basili, one of the software engineering gurus "challenged" the software engineering community asking them to grow up: apply scientific principles to software engineering. This statement applies now for too many SPI experts taking things from book as granted without thoughts and wise interpretation.

To get rid of this theoretical reputation, activities related to business issues have to be performed at given point in time along a software process improvement program: at the starting point of the initiative, during the assessment, during the action planning phase and periodically during the implementation. Some strategies have recently come up i.e. the SPICE process improvement guide [PIG95], the SEI Ideal model [PeR94], the ISPI Action Focus Assessment [CKP96] [TAK96] and the **ami** approach (goal and metrics driven) which combined with CMM assessment has proven some efficiency. The later will be discussed next

AMI® in the context of structuring SPI PROGRAM

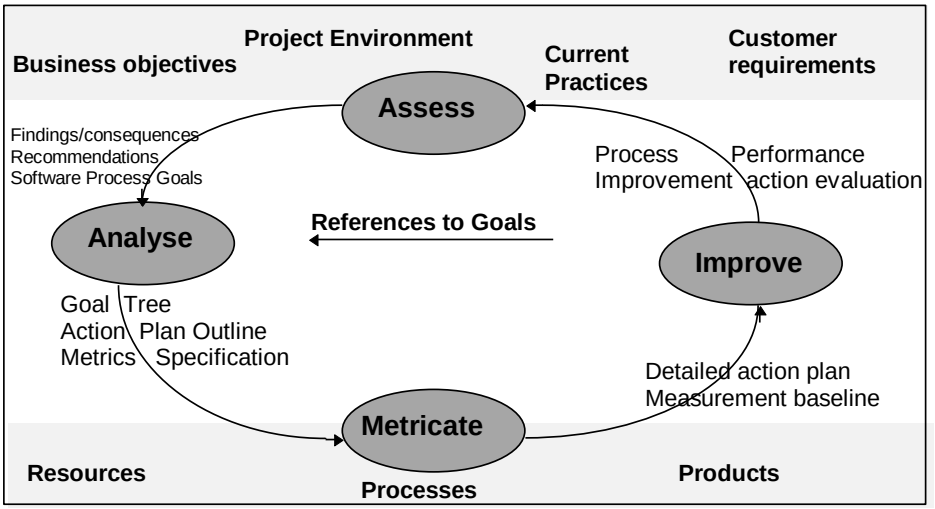


Figure 1: ami® **activities in SPI context**

Origin.

ami® (application of metrics in industry) was a two-year project which started in December 1990 under sponsorship of DG XIII of the Commission of the European Communities through the ESPRIT program promoting the use of measurement in software development. The goal of the project was to develop a practical approach for implementing software measurement and to validate it on a variety of projects all over Europe [DLS93]. The approach is described in the **ami®** handbook [AMI95]. The **ami®** paradigm (Assess, Analyze, Metricate, Improve) is

similar to the Shewart cycle (plan, do, check, act) for process improvement (refer to Figure 1). **ami®** has taken this cycle, based on common sense principles, and developed it for software measurement. The **ami®** method is a stepwise, iterative, incremental, goal-oriented procedure coupling together a model-based process assessment technique with a quantitative approach to software development issues from the viewpoint of the process, product and resources. **ami®** has been extensively described in [Deb94].

ami® applicability in SPI

The main bottleneck in the majority of SPI programs has been the strategy to adopt after running a capability assessment. The **ami®** concepts and framework (Assess, Analyze, Metricate, Improve) although defined for implementing software measurement, can be easily applied for structuring improvement program: goal-driven, supported by metrics. As a matter of fact, measurement should play a major role in any SPI initiative:

- Provide a quantitative basis of comparison for future changes
- Help in better understand the issues that may or may not have been identified in the qualitative analysis (assessment) e.g. high cost of certain activities may make the priority improvement targets
- Make decision making process less risky.

The 4 **ami®** activities are tailored next in the context of an SPI program

Assessment

The first step deals with the assessment of the software development environment for defining software process goals. Weaknesses and critical parts of the software development process are first pointed out. From those findings, consequences and business objectives as emphasized by the management, software process goals are defined (step 2). A hierarchy of top level goals is being considered depending on the maturity of the development process. Before setting up "change" goals (e.g. to improve productivity while maintaining quality), one should envisaged "understanding" goals e.g. support of estimation process with historical data). The assumption behind improvement goals is that the process is well defined. Then a validation of goals against the assessment conclusions, the timescale and the dedicated budget is performed to avoid too ambitious goals to be set up (step 3).

Analyze

The aim of the second activity is to build a so-called goal-tree to visualize the relationships between business objectives, software process goals, high level improvement activities and related follow-up metrics. The process is based on the Goal/Question/Metric paradigm [BaR88]. When software process goals are defined (e.g. make reliable estimates, achieve full traceability), one should try to identify viewpoints (who is a playing a role in achieving this goal) and entities they manage in this context (e.g. estimation process, estimation review). For

each viewpoint, questions on the entities in the context of this goal will be set (e.g. how accurate is the estimation process, are concurrent estimates performed). Then software process goals can be broken down into sub-goals that may be improvement activities. This process is performed until single improvement actions that can be instrumenting through metrics are reached. After having verified the consistency of the tree (step 5), metrics are derived at all levels of the tree with the help of questions (step 6). The results are a documented goal tree and the associated set of metrics. With some background information, an improvement action plan outline can de produced.

Metricate

The metricate activity encompasses three steps:

- Writing the measurement plan which is the reference document for collection and analysis of data and for ease of tracing of these tasks (step 7)
- Collecting the data (step 8)
- Verifying the data (step 9)

Several types of metrics can considered here:

- Overall performance/efficiency metrics to establish a measurement baseline to measure impact of actions on daily business
- Metrics to be used for root-cause analysis of the problem, and consequently the establishment of a detailed action plan.
- Metrics to follow-up closely the piloting of new process.

Improve

The exploitation of measures has to be performed in reference with the goals defined in the analyze activity. Improvement activities are implemented and follow-up by metrics with regards to impact on business, project and/or process performance.

Case study 1

This case study only covers the assessment and action planning phase. The **ami®** concepts like goal tree have been applied later in the process to overcome the issues of senior management commitment. The following four main lessons learned will be illustrated along this case study:

1. Involve management upfront by having them investigating business and software related goals as well as critical areas.

2. After the assessment, start building a goal tree together with the main actors based on both inputs of management vision and assessment findings.

3. Do some initial root-cause analysis driven by metrics on problems before doing detailed action planning.

4. start to attach upfront metrics to the different levels of goals to collect baseline measurement for measuring changes.

Assessment

The decision to have a software development capability assessment based on the CMM framework came from corporate management who initiated a company-wide software process improvement program. The assessment was a three-week exercise (refer to Figure 2), 2 weeks of interviews and derivation of findings and initial recommendations, 1 week where the assessment team together with the SEPG sets the ground for the action planning.

No surprise. The assessment revealed weaknesses in most of the management practices from level 2. At the end of the second week, the results were presented to the organization. This was followed by a so-called executive session with the management team to discuss the results and set priorities. This session is essential since driving the 3rd week contents. The team (composed of members of the local organization and external and corporate consultants) started to ask questions about priority areas (project, findings) to concentrate on as well as on business objectives. Unfortunately, the team was not successful for collecting these information. The management was not prepared to this session (just a theoretical introduction to SPI some weeks ago) They just have been overloaded with a bunch of problems, They did not have the necessary time to evaluate in depth the situation. The only message (driven by consultants) was to concentrate on level 2 issues like project management and requirements engineering.

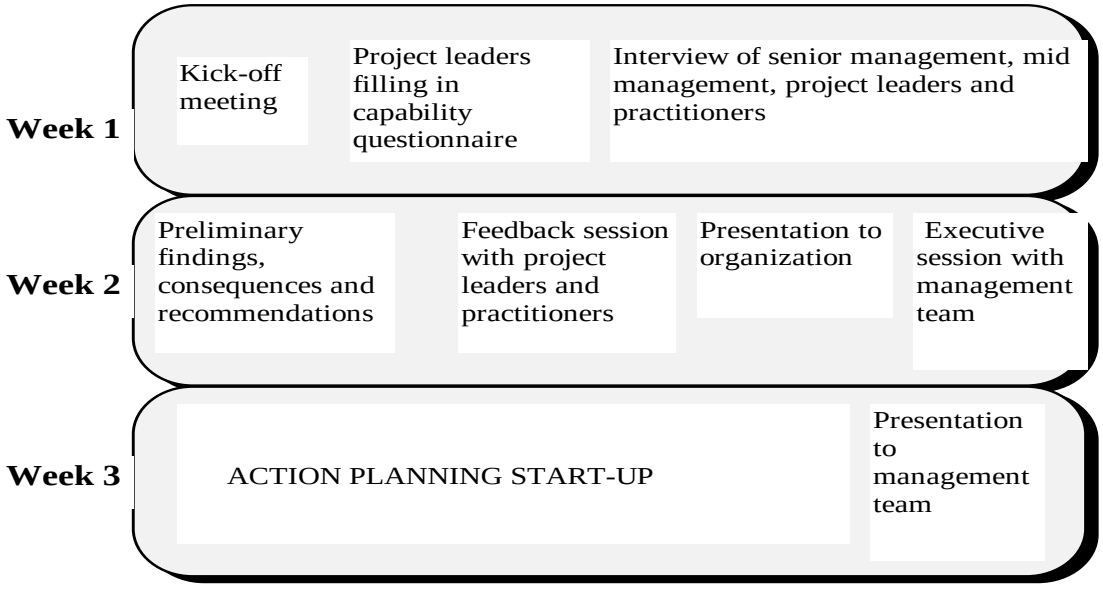


Figure 2: Schedule of assessment

- ✍ Lessons learned 1: The management even exposed to SPI concepts should have been briefed upfront what they have to do concretely during the assessment, before and after. The "ideal" talk from SPI suppliers did not bring them to think in terms of business objectives, critical areas, that are the customers of the SPI project. If the last project failed to deliver the product on time due to a deficient configuration management system, this should drive the forthcoming actions towards the current release. If a platform development starting up now is critical for the survival of the organization, then the emphasis should be first in requirements engineering of this project!! SPI should not be viewed as an independent team. Furthermore, business management should be involved.

Due to this low input from the third week has been more or less a theoretical exercise. Findings were translated into goals for the working groups, consequences into so-called expectations (what to expect if the goals are met), and finally recommendations into implementation steps following a generic scheme (training, collect data, analyze data, devise new processes, pilot, deploy). Even some "guestimates" of cost and schedule were also provided. At the end of third week, the management was presented with the results and got hungry. Their impression was that the team has done a rewriting of the first presentation (actually partially true) with little added value. Their expectations might have been too high as well.

Two lessons could be drawn out:

- ✍ Lessons learned 2: Bringing the management to show a rough draft of action plan without having investigated the details of the problem was not appropriate. What could we expect from management? confirmation that we were on the right track?: not with the level of information that was provided. The link to business goals was completely missing, a list of improvement goals per KPA (Key Process Area) were available when problem-oriented goals would have been more appropriate.
- ✍ Lessons learned 3: Providing an overall schedule & effort estimation for the next year was very much doubtable. In a software project, it is of usual practices to start with a high level plan and effort estimation and then as soon as requirements gets further analyzed, do some detailed planning. The authors are not sure that this approach is also fully applicable for SPI project. At the beginning, requirements are much too vague and no history is available. A better strategy would be first to differentiate across the issues what is from the first sight short, mid or long term and the associated level of effort (low, mid, high, depends on analysis). Then for each major finding, a detailed root-cause analysis should be performed (if necessary further collection of metrics), describe the target process or vision (where do I want to be) and then do a detailed planning for the next 3 to 5 months and some high level milestones for a one-year period.

Analyze

Something had to be done very quickly, otherwise like 2/3 of the worldwide initiative, it will fail to provide impact on the organization early enough to keep momentum and consequently organizational commitment. Corporate experts examined the current status of the program. The results of the action planning week needed to be reengineer in a goal tree starting with

business goals. Business management was brought together with the R&D management in a brainstorming session to define business goals and brainstorm initial software process goals. The current concerns of the president of the division were twofold:

- Decreasing quality and delays have caused customer unsatisfaction and penalties.
- A critical project to recover customer confidence who threaten to reduce the market share, was starting.

Three mid/long term goals were agreed among the management team, dealing with quality improvement and reduction of delays through better estimation and visibility on progress. A short term goal was raised as well, making the starting project "Guinea Pig", the first customer of the SPI initiative.

A first priority list for SPI goals was produced by ranking assessment findings according to previously defined business goals. The president of the division emphasized typical management practices like risk management and a better relationships between marketing and R&D department. Players or viewpoints for the second round were also selected per business issues:

- Quality: R&D management, marketing, system, QA, SEPG, Integration/validation team
- Project management: R&D management, planning, SEPG

For each session the "Guinea Pig" project leader was made available. The sessions run as follows:

- Definition of priority software process goals through identification of major entities and related deficiencies which improvement would contribute to the business goal.
- Write down a set of questions that clarifies the current status qualitatively and quantitatively (data collection phase followed the 1st session).
- Derivation of high level improvement actions.
- Validation of the resulting goal tree and prioritization of actions with the project leader: *Does it make sense for his project and when*?

For instance, for the quality goal:

The major entities were marketing requirements definition, technical requirement definition, review of requirements, validation test definition. Some assumptions were made and had to be verified after the session:

- Majority of major defects due to uncomplete or untestable requirements

- Not enough time spent in initial project phases compared to industry average. Exit criteria was not formalized.

Those assumptions were documented with questions and metrics to collect. After all, two software process goals were selected, dealing with feasibility of marketing and technical requirements. Typical actions were related to approach the requirement phase in a more formal way (numbering, fields like performance, ...), applying inspections on requirements documents, formalizing exit criteria including having full validation test cases defined, ...

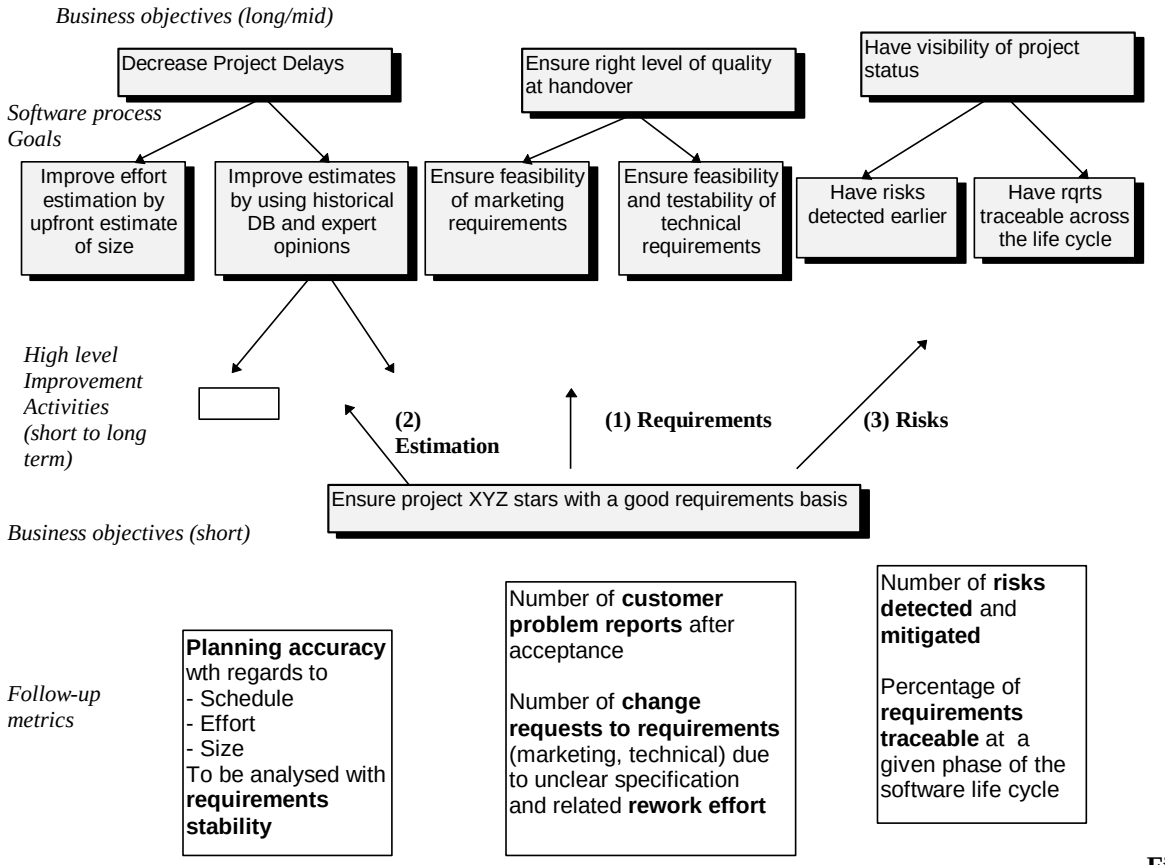


Figure 3: Goal Tree

After several session and data analysis, a final goal tree covering business objectives, related software process objectives, high level actions to implement and metrics to check that the objectives are met (Figure 3) was built and presented to management for approval. Finally the management could obtain a clearer view of the program, how it stands towards daily business. Now the SEPG and other experts could go through detailed action planning in parallel with the implementation of some “quick fixes” that would give visibility of the program to the whole R&D community. An important aspect was to tackle the investigation not around KPA but around problem areas. The KPA approach although useful during the assessment is too restricted during the action planning, hiding cross-KPA issues (e.g. traceability).

✍ Lessons learned 4: Metrics were defined at the same time than the goal. Then collection mechanism could be put in place rapidly. The management decided to refine goals with

quantitative targets. Due to the variety of projects, those targets were set up for each major product range or markets. This was more appropriate than setting up objectives for the whole organization which would be longer to achieve and consequently less motivating for the project staff.

Bringing back the SPI program on track had to do with linking it to daily business issues: business goals, measurement of performance to evaluate changes, taking critical projects as SPI customers. The measurement program will allow to keep the SPI initiative alive together with continuous management follow-up. A more detailed action plan is now to be worked out and implemented alike any software project.

Case Study 2

The second case study is an interesting one in the sense that it demonstrates that the **ami®** approach can really solve the problem of deriving efficient actions. It also highlights the role of linking SPI to business goals.

The company is a defense market actor of one of the European countries (for confidentiality reasons we are not able to give more precise details and even if authorized, more details will not add value to the example). The software development staff is about 70 people, less than 10% of the total number of employees.

Software is one component of the systems which encompass several other specialities. The company started a process assessment according to the CMM Model in 1994 and was assisted in doing that by one of the most famous US authorized organization. All the participants reported about the professional features of the assessment instruments used and about the skills of the assessors. To follow up with the assessment results, a brainstorming seminar was then organized by the company in order to formalize action plans addressing the main weaknesses observed. This was organized within a 3 months delay which was fitting one of the key success factors of SPI.

But despite this efficient start and some observable improvement by the end of 94 (new procedures in place), the SEPG was not completely satisfied and the senior management was slightly loosing confidence in being able to achieve significant progress without a more structured and tacked approach. At this stage discussions were initiated on how **ami®** can overcome the problem and the final decision was made on the following arguments :

- **ami®** helps in defining quantitative achievable process targets; furthermore, when they have been defined, the G/Q/M concept embedded into the method ensures that, at any stage, the tactics are aligned with the strategy,
- detailed actions are derived in order to match initial goals and be cost effective rather than being CMM compliant,

- indicators are defined at the same time to monitor actions and continuously track progress towards targets; interim assessment might be skipped without trouble.

CMM training and assessment based on SEI method (SPA or CBA-IPI) produce consensus among the whole organization on what the major process bottlenecks are, in CMM terms, to achieving the business goals.

Agreement from every level and all groups within the organization on what process improvements to undertake systematically is gained. These are the improvements that would benefit quality, productivity, and timeliness most directly as seen by the people who know the current process.

Extensive use of the G/Q/M paradigm as described in the **ami®** method is cost effective to succeed in defining actions. Steps 4/5/6 of the method (From goals to sub-goals, Verifying the goal tree, From sub-goals to metrics) are applied combined to the assessment findings:

Each class of participant (to whom 1 or more sub-goals have been allocated), reviews the assessment findings and answers the following questions:

- Is there a relationship between the sub-goal and the main process weaknesses observed?
- What type of activity could we implement to reach the sub-goal? Is it a short term or mid/long term corrective action?
- What type of action would be effective and efficient in performing sub-goal?
- What type of metrics data would show progress on the action and toward the sub-goal?

As soon as the list of actions is established and validated against the business goals, the complete plan may be prepared. Effort, schedule and owner of each action has to be identified, all figures have then to be compiled before achieving a final check against business goals, overall budget and time frame initially fixed. Priorities may still have to be put in order to fulfill SPI budget constraints. Detailed action plan is completed with the list of metrics to track the goals achievement; for each metric, an exploitation schema is provided i.e. some scenarios to react in case of deviations observed from the initial target.

The above initiative conducted from fall 1995 up till now has largely demonstrated these benefits.

Assessment

Because the CMM assessment had already been achieved prior to our work, assessment was centered on the primary goals definition for SPI. Our consultant met about 10 key people in the organization structure according to a guided schema to understand and later formalize :

- the role of the software component in the market strategy and the customer’s perception of it,
- the reasons for initiating SPI : expectations, links with mid term organization strategy,

- the quantification of the targets.

Along the interviews, 3 main observations were made :

- one year after the CMM assessment some priorities for improvement were changed,
- the understanding of the key players differed regarding the role of software in the business;

This type of interviews raise discussions which help to prepare the changes in the process and sensibilise on the CMM concepts. This phase will, by the end, be considered as a good catalyst to enable SPI.

Analyze

The analysis phase was dedicated to the decomposition of primary goals "reduce software development cost" and the definition of the main process evolutions. Reduce software development cost was split into a mid term goal "match the initial budget" —which is to a certain extent already decrease of overall cost— and into a long term goal which was considering a real decrease of the initial budget for similar systems. This decomposition resulted into :

Reduce development Cost

1. Manage the stability of requirements

1.1 Reduce the number of change requests

2. Acquire the knowledge of costs

2.1 Analyze the reliability of initial estimates
2.1.1 Identify the deviations and origin

2.2 Identify how the effort is split along the life cycle phases
2.2.1 Define effort /phase
2.2.2 Identify rework effort
2.2.3 Identify the availability of resources

For each sub goal, actions were defined based on the assessment results. Let's take sub goal ; " Acquire the knowledge of costs" as an example.

Following the assessment, it was decided to refine the estimation procedure on the basis of a collection of past experiences: this refinement addressed the definition of the structure of the repository for experiences and some mechanisms to reuse this experience. A second action was initiated for the tracking procedure. Therefore it was decided to help software leaders in their analysis of the deviations in order to reduce the risk of not adapted or no corrective actions at all.

The last step of this phase was the definition of indicators. Continuing with the same example, the SEPG came out with the following indicators:

- the deviation rate for an homogeneous group of projects (Figure 4)

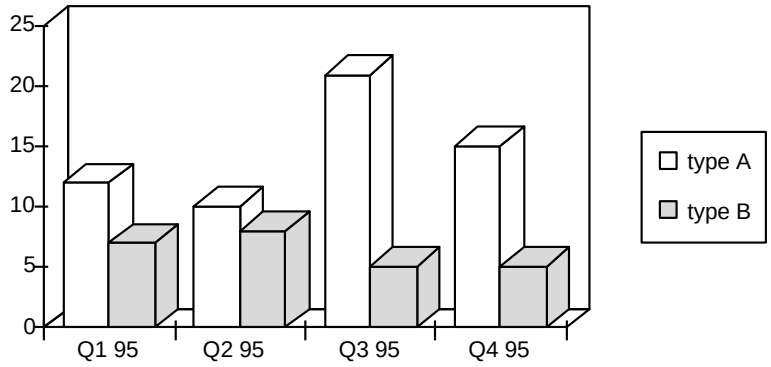


Figure 4: Deviation rate

- the deviation rate — for a specific project — observed at each phase of the development (Figure 5).

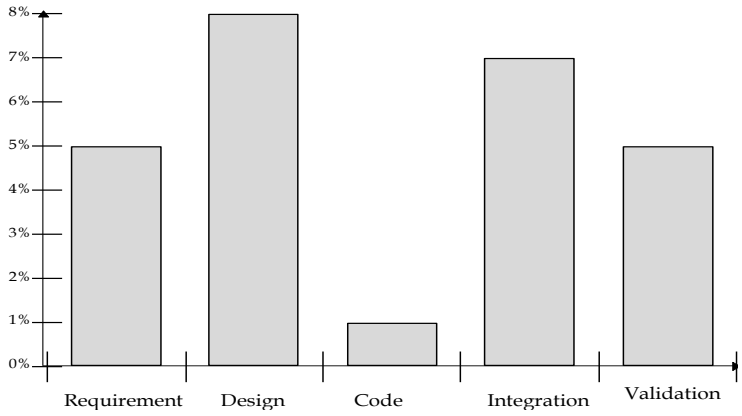


Figure 5: Deviation per phase

These two indicators were extremely helpful to give visibility in the deviations for projects and to initiate analysis of the reasons for deviation: external occurrence or unrealistic initial estimate or difficulty not forecasted etc.

Metricate

The improvement/measurement plan being completed, these metricate stage could start. The plan was first deployed on 3 pilot projects to validate the mechanisms for data collection and exploitation. The observation period was 3 months with some data collected regularly, some collected when reviews were organized at the end of a development phase and some by the end of the project (this last category has not been verified during the prototype phase). Together with the mechanisms we wanted to consolidate the budget estimation for the improvement and measurement.

This pilot phase was also judged very positive as far as it raised motivation among the participants who then communicated their enthusiasm to the rest of the software community. Some minor adaptations were necessary in the measurement plan in order to get a completely adapted collection mechanism and to avoid duplication with the existing global system for cost management.

Conclusions

This experiment proves that there is a real need in assisting SEPG in deriving action plans from the assessment results. It is not an easy task and without a step by step method there is a big risk of deviation and, at the same time, to loose all the buy in raised during the collaborative work of the assessment. The other main risk is to be too strongly guided by the model and to forget the link with the goals for improvement.

Many organizational units are looking for a certain maturity level by a given time and if you ask to some of the key players the justification for it, he/she will be unable to find a reasonable one. Furthermore, senior managers are usually not motivated by a maturity level but by more economical factors linked to the business.

Therefore, a translation is necessary between business goals and process improvements which will hopefully result into measurable products benefits. This translation is straightforward when the **ami®** approach is used.

Conclusion

Those two case studies have shown the necessity to link any software process improvement program to business goals and to follow it up closely with indicators related to the initial goals. Those simple principles have been so far seldom followed by the SPI actors. One reason lies in the lack of emphasis on those issues from SPI suppliers. It might be also related to the inherent characteristics of engineer to tackle any problems as it was a technical problem. SEPGs, Working groups are composed of technical people that have been awarded to low or mid-management position according to their technical results. At level 1-2 of the CMM layer, we are dealing with management-related issues: estimating, planning, controlling, tracking, roles and responsibilities, commitment, ... Maybe that does not excite engineers enough?

References

[AKC96] Kuntzmann-Combelles A., from assessment to improvement actions: compared experiences with CMM and SPICE, to be published at *the 5th European conference on software quality*, Dublin, Sept 1996.

[AMI95] Pulford K., Kuntzmann-Combelles A., Shirlaw S.: A Quantitative Approach to Software management, ISBN 0201877465, 1995 , Addison Wesley

[BaR88] Basili, V., Rombach, H.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Transactions on Software Engineering, Vol 14, Number 6, June 1988.

[CMM93a] Paulk M C, Curtis B, Chrissis M B: Capability Maturity Model for Software, version 1.1, CMU/SEI-93-TR-24, February 1993.

[CMM93b] 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.

[CKP96] Cox C., Kasse T., Pinney D.: Guidance for Action Planning In*: Proceedings of the SEPG 96*, Atlantic City, New Jersey, May 1996.

[Deb94] Debou C: ami a new paradigm for software process improvement, In: *Proceedings of the first ISCN Seminar*, Dublin, May1994

[DFH95] Debou C, Fuchs N., Haux M.:ami: a Taylorable Framework for Software Process Improvement, In: *Proceedings of the second ISCN Seminar*, Vienna, September1995

[DKR94] Debou C, Kuntzmann-Combelles, Rowe A.: A quantitative approach to software process, In: *Proceedings of the 2nd international symposium on software metrics*, Oct 1994, London.

[DLS93] Debou C, Lipták J, Shippers H.: Decision making for software process improvement: a quantitative approach, In: *Proceedings of the 2nd international conference on "achieving quality in software" ACQUIS 93*, Venice (Italy), pp 363-377, Oct 1993.

Also In: *The Journal of Systems and Software*, 1994; 26:43-52, Elsevier Science Inc.

[Dym95] Dymond K. A guide to the CMM, Process Inc US, ISBN 0-9646008-0-3, 1995

[HMK94] Haasse V, Messnarz R, Koch G., Kugler H, Decrinis P: Bootstrap: Fine-Tuning Process Assessment, In: *IEEE Software* , pp25-35, July 1994

[Hum89] Humphrey, W.: Managing the Software Process, Addison-Wesley, Reading, Mass., 1989.

[Kas96] Kasse T.: Improving the Action Focus of Software Process Assessments, In: *Proceedings of the SPI '95* Conference, Barcelona, Spain, November 1995

[Kit96] Kitson D.: An Emerging International Standard for Software Process Assessment, In: *Proceedings of the European SEPG Conference*, Amsterdam, June 1996

[PeR94] Perterson B, Radice R.: IDEAL: an integrared approach to software process improvement (SPI), In: *Proceedings of SEI Symposium*, Pittsburgh, August 1994.

[PIG95] SPICE consortium: Guide for Use in Process Improvement, ISO/IEC

[SEPG95] Proceedings of the SEPG conference, Boston, May 1995, obtainable from the Software Engineering Institute.

A Minimum Set of Metrics for Effective Process Management

Eric Trodd

Brameur Ltd
Saracen House
232 Fleet Road,
Fleet, Hants GU13 8BY
England
Tel: +44 1252 812252
Fax: +44 1252 815702
Email: 100142.3334@compuserve.com

Abstract

One of the key objectives of effective process management is to ensure that the results of any changes made may be measured to verify that they have resulted in improvements to performance and quality. This requires the establishment of a set of key performance indicators (KPIs) which are related to business goals as well as an associated set of project performance measures which provide the quantitative information from which the KPI values are determined. The KPIs can then be used to help identify opportunities for improvement and to quantify the improvements required.

This paper discusses the practicalities of establishing a measurement programme to support effective process management related to software development and maintenance, based on a simple hierarchy of method-independent KPIs for software development, a set of management-related questions which the KPIs can answer and 20 easy-to-use project measures which use quantitative information commonly available from such projects. Models for combining these measures to provide the answers to the questions and to demonstrate achievement of KPIs are outlined.

This approach will assist a software measurement programme to be used successfully as a management tool to identify improvement opportunities which are linked to management goals. It is based on work done in developing the METKIT training package, part of an ESPRIT project to develop educational materials on the use of measurement in software engineering.

Lastly, the compatibility of the approach with international models such as CMM and SPICE is discussed.

Introduction

The approach to implementing a measurement programme to support software process management described in this paper was developed as and extension to the METKIT project, partially funded by the Commission of the European Communities as ESPRIT project 2384. This project developed a set of 20 training modules on process improvement and "Measurement as a Management Tool" **[Reference 1]**. Also a Measurement Starter Kit was developed to provide direct assistance to organisations wanting to set up a software measurement programme **[Reference 2].**

The approach is intended to generate quantifiable information of particular use to management. Its philosophy is that measurement should provide quantifiable information aligned to business goals. The METKIT approach is derived from leading companies in Europe, USA and Japan who have already used measurement successfully.

The Fundamental Questions

To manage its business processes successfully, an organisation needs to be able to answer a number of fundamental questions - to ensure that relevant and useful programmes are implemented:

- **Where are we?** - what is our current capability believed to be
- **Where do we want (or need) to go?** - what improvements are desired (or required) from the current level of capability
- **How do we get there?** - what extent of change will be necessary to achieve any improvement
- **Have we got where we wanted to be?** - will the programme help to demonstrate that desired improvements have been achieved?
- **How do we compare against the competition?** - will the programme allow benchmarking of performance and capability against industry standards and other organisations?

This requires the establishment of a set of key performance indicators (KPIs) which are related to business goals as well as an associated set of project performance measures which provide the quantitative information from which the KPI values are determined. These can then be used to assist in identifying opportunities for improvement and to quantify the improvements required and achieved.

Software development and maintenance may be regarded as a "superprocess" and meaningful metrics are the major key to their effective process management and improvement.

The questions listed are also the basics for setting up any measurement programme to support process management and should be approached in the same sequence. If measurement cannot indicate current performance ("where are we now?"), it cannot demonstrate that improvements have been achieved or provide a basis of comparison as part of a benchmarking exercise.

Problems and Barriers

To establish an effective measurement programme, a number of problems and barriers to progress will need to be tackled, such as:

- lack of management commitment
- imprecise or inappropriate objectives
- unclear linkage to business and management goals
- differing management and staff expectations
- assessment criteria not defined
- lack of feedback
- concerns about misuse of information
- cultural change needed
- willingness to continue in the longer term

Any one of these problems and barriers can cause the programme to fail - not just functionally, but in terms of management success. The programme may generate information, but it does not get acted upon. *Ensuring that the programme and its objectives are clearly linked to business and management goals is the key barrier to be tackled*.

While management is unlikely to object *per se* to any initiative to improve software development performance, the key to success is having an "Executive Sponsor" - a senior manager, preferably at board level, who is prepared to support its defined goals, take overall responsibility for the programme,

provide sufficient resources and act as the focal point for communication in both directions between the programme and senior management.

An iterative approach to a measurement programme

Any approach to measurement will need to be iterative, since it is very likely that the programme will need adjustment after its first period of operation - also management goals change with time and the projects to which the programme is applied will change.

Figure 1 represents a recommended structure for setting up and maintaining a measurement programme. Important steps are devising success criteria both for the programme as a whole (and agreeing then with the executive sponsor) and establishing the individual KPIs and measures from projects - also establishing a feedback mechanism from which the scope and goals of the programme will be validated and if necessary adjusted.

Figure 1. The Basic Steps - An Iterative Approach

1. Establish Scope and Goals
   - appoint an executive sponsor
2. Define KPIs, Measures and Models
   - devise success criteria
   - explain the programme
3. Set up Data Collection Infrastructure
   - develop feedback mechanism
4. Set Base Lines and Targets for KPIs
   - perform management reporting

Establishing a Measurement Programme

The rest of this paper explains in more detail eight major steps of this approach, with particular emphasis on Step 3, which establishes the KPIs, measures and models for analysing the measurement information obtained from project work.

Step 1.Appoint an Executive Sponsor

From the business point of view, this is the key step - without an effective sponsor the programme is unlikely to have a long term future. The sponsor should be a *leader* who is:

- committed to change
- dynamic
- a strong motivator

and who will set challenging but realistic targets. These targets refer to both the establishment of the measurement programme (e.g. obtaining measurement data from all development projects within 6 months) and, even more importantly, to the business goals that it achieves (e.g. demonstrating a 10% improvement in productivity or a 50% reduction in delivered software faults).

The other key player is the measurement programme leader or manager, who will be responsible for establishing and running the programme and achieving the targets agreed with the executive sponsor.

Step 2.Establish Scope and Goals

This step determines, with the approval of the Executive Sponsor, the scope and goals of the programme. The practicality of these goals can be demonstrated by the programme's ability to provide the answers required on the basis of quantitative, objective measures.

Examples of practical, business-related goals include:

- assess the productivity of development projects
- identify where changes can improve productivity
- assess the effectiveness of reviews and testing
- determine the effort spent on rework
- identify trends in defects
- determine what factors affect the accuracy of estimation

It is important that the programme is clear on the extent to which these goals are long term or short term (or even one-off). If the ability of a measurement programme to deliver meaningful results is uncertain, then the only viable goals may be the first two - and without any specific commitment by management to make any changes. Once these goals are found to be viable, then the goals can be respecified for a second iteration of the programme, (e.g. reduce error rates by 50% within 12 months from the initial levels detected by the programme).

Step 3.Define KPIs, Measures and Models

The measurement programme needs to set up an initial framework which can be extended and enhanced as the programme develops and as business goals change. The following steps represent a process for first establishing a measurement programme and subsequently extending the programme's set of KPIs, questions, metrics and models:

- define Key Performance Indicators (KPIs) *in line with current Business Goals and Needs*
- develop a set of questions relating to the KPIs so that you can measure
    - where you were
    - where you are
    - how you have improved
- define metrics to capture the relevant data
- develop models for combining the metrics that enable the questions to be answered

A sample set of initial KPIs, questions, metrics and models is described below.

Key Performance Indicators

Figure 2 provides an example set of KPIs relating to business goals of productivity, quality and estimation. These KPIs have been chosen since they address issues of direct concern to the business and hence senior management.

Figure 2. Key Performance Indicators

Productivity	Function Points per person month Cost per Function Point Count
Quality	percentage effort spent on rework percentage effort spent on reviews effectiveness of reviews effectiveness of systems testing average effort to correct defects found during

	development percentage bad fixes during development
Estimates	Estimated vs. actual effort (variance) Estimated vs. actual timescale (variance)

(Function Points can be replaced by lines of code or other recognised measures of software size)

These KPIs are intended to be both independent of any specific development method (including life cycle model) and capable of being determined by simple measurements made during project work. The only pre-requisites for their usage are:

- software output is measurable (in this example as a Function Points count)
- reviews are performed which are capable of detecting and recording errors
- the time and effort for work performed by projects is estimated first
- basic project records are obtained for time and effort spent, defects found and rework

Any software development process which meets the requirements of ISO9001/TickIT or corresponds to Level 2 of the CMM model for process capability should satisfy these pre-requisites and so should be able to generate these KPIs.

Typical Questions

Current Performance

Figure 3 below lists a set of questions which address the KPIs identified above (see Figure 2). These questions focus on current performance, prior to any attempt at improvement. Quantitative answers to these questions, which describe the current situation and level of performance, are derived from detailed measures obtained from project work (described later)

Each question is uniquely numbered (Q1.1 to Q1.10) to provide traceability to the individual project measurements and to each KPI.

Figure 3. Typical Questions

Productivity	Q1.1 What is the productivity of the projects? Q1.2 What is the cost of productivity?
Quality	Q1.3 What percentage of effort is spent on rework? Q1.4 What percentage of effort is spent on reviews? Q1.5 How effective are reviews? Q1.6 How effective is systems testing? Q1.7 What is the average effort to correct defect found during development? Q1.8 What is the percentage of bad fixes during development?
Estimates	Q1.9 How accurate are project effort estimates? Q1.10 How accurate are project timescale estimates?

Improvement Related

A further set of questions (Q2.1 to Q2.6) linked to the KPIs can be used to analyse the need for improvement based on current and recent performance. Like the performance-based questions (Q1.1 to Q1.10), the answers will de derived from the set of 20 measures applied to project work.

Figure 4. Further Questions (improvement related)

Productivity	Q2.1 What factors affect productivity?
Quality	Q2.2 Which development phases generate most defects? Q2.3 What percentage of defects found during development are corrected? Q2.4 Which modules cause most operational failures? Q2.5 What is the breakdown of effort across development phases?
Estimates	Q2.6 What factors affect the accuracy of estimates?

Commonly Used Measures

Once the required KPIs have been identified, together with the associated questions, a set of quantitative measures to be provided by projects can be defined. By cross-referencing the measures to the questions (and hence the KPIs), a minimum set which is sufficient to satisfy the needs of the measurement programme can be defined. (Some of these measures will relate to more than one question.)

Figures 5 and 6 provide a sample set of basic measures (M1 to M20). These measures should be readily obtainable from any project which has processes in place for estimating, planning reviewing and recording progress.

Measures M1 to M11 (Figure 5) focus on product size and development time and effort.

Figure 5. Commonly Used Measures - 1

M1	Size (Function Point Count, KLOC, etc)	Q1.1, Q1,2
M2	Development Cost	Q1.2
M3	Actual Effort	Q1.1, Q1.3, Q1.4, Q1.9, Q2.5
M4	Estimated Effort	Q1.9
M5	Actual Phase Effort	Q1.9, Q2.5
M6	Estimated Phase Effort	Q1.9
M7	Actual Elapsed Time	Q1.10
M8	Estimated Elapsed Time	Q1.10
M9	Effort spent correcting defects during development	Q1.3, Q1.7
M10	Effort spent on reviews	Q1.4
M11	Effort spent correcting defects during first 3 months of operational use	Q1.1, Q1.3, Q1.9

Measures M12 to M20 (Figure 6) focus on defect introduction and removal.

Figure 6. Commonly Used Measures - 2

M12	No. defects introduced during a phase	Q1.5, Q2.2
M13	No. defects detected during a review	Q1.5
M14	No. defects detected during a phase	Q1.6, Q2.3
M15	No. defects present at review from earlier phase	Q1.5
M16	No. defects corrected during phase	Q1.7, Q1.8, Q2.3
M17	No. defects caused by bad fixes	Q1.8
M18	No. defects per module found during operational use (per period)	Q2.4
M19	No. operational faults found attributable to development (per period)	Q1.6
M20	Size of module	Q2.4

Those measures which are phase related (M5, M6, M12, M14, M15) should be obtained for each project phase, particularly where a standard life-cycle model is used for projects. This will help to allocate defects to phases and hence identify some of the factors concerned with quality (see Q2.2 above)

Example models for answering questions

Quantitative answers to the questions (and hence KPI values) can be provided by combining the measures M1-M20 as appropriate. Figures 7, 8 and 9 below provide a set of example models for doing this.

Figure 7. Example Model for Answering Questions - 1

Q1.1	Project Productivity	$\frac{\text{Size M1}}{\text{Actual effort M3}}$
Q1.2	Cost of Production	$\frac{\text{Size M1}}{\text{Development cost M2}}$
Q1.3	% effort spent on rework	$100 \times \frac{\text{Rework effort M9}}{\text{Project effort M3}}$
Q1.4	% effort spent on reviews	$100 \times \frac{\text{Review effort M10}}{\text{Project effort M3}}$
Q1.5	effectiveness of reviews by phase	$\frac{\text{No. defects detected M13}}{\text{No defects introduced M12 + No. defects present from earlier phase M15}}$
Q1.6	Effectiveness of System Testing	$\frac{\text{System Testing Defects M14*}}{\text{All Testing Defects M14** + Faults M19}}$
Q1.7	Effort to Correct Defect	$\frac{\text{Rework Effort M9}}{\text{Defects Corrected } \sum\text{M16 ***}}$
Q1.8	% Bad Fixes During	$100 \times \frac{\text{Defects caused by bad fixes M17}}{}$

	Development (Defect Propagation Ratio)	Defects Corrected $\sum$M16 ***

Figure 8. Example Model for Answering Questions - 2

Q1.9	% Accuracy of project effort estimates (Effort variation) (Phase effort variation)	 100 x $\frac{\text{Project effort M3 - Estimated effort M4}}{\text{Project effort M3}}$ 100 x $\frac{\text{Actual effort M5 - Estimated effort M6}}{\text{Actual effort M5}}$
Q1.10	% Accuracy of project timescale estimates	100 x $\frac{\text{Actual time M7 - Estimated time M8}}{\text{Actual time M7}}$

Figure 9. Example Model for Answering Questions - 3

Q2.2	Defect-prone phases	$\frac{\text{Phase defects introduced M12}}{\text{Total phase defects introduced } \sum\text{M12}}$
Q2.3	% Defects corrected	$100 \times \frac{\text{Total phase defects corrected } \sum\text{M16}}{\text{Total phase defects detected } \sum\text{M14}}$
Q2.4	Modules with most operational failures	$\frac{\text{Operational module defects M18}}{\text{Module size M20}}$
Q2.5	% Effort across phases	$100 \times \frac{\text{Phase effort M5}}{\text{Project effort M3}}$

Q2.1 and Q2.6 cannot be answered by a simple model, partly because time will be needed to build up a database of project data sufficient to start an analysis. Typical factors will contribute to Q2.1 (productivity) are:

- project timescales
- amount of rework
- accuracy of project size estimates
- complexity of project
- time for systems testing
- time for analysis and design
- unplanned changes to requirements
- novelty of project
- size of project team
- experience of staff
- stability of project team
- use of new tools and methods

While many of these factors can be expressed in terms of the measures M1 to M20, a few of them (e.g. novelty of project, use of new tools and methods) are difficult to express as a measure. At this point more sophisticated estimation models can be applied which can take into account the likely impact of these harder to quantify factors - in effect these tools make available data and analysis from the software industry as a whole rather than just the organisation which is setting up the measurement programme.

Step 4.Establish a data collection infrastructure

A consistent and reliable means of collecting project measures is needed. The measurements M1 to M20 are intended to be already available from projects, and so the work needed is mainly collecting "off-line" from existing project records and analysing the data. This minimises work for projects and therefore maximises the likelihood of their co-operation.

The programme infrastructure should enable the following questions to be answered:

- who collects data?
- when is it collected?
- where is it collected from?
- how is it collected and validated?
- who analyses and presents the results?

Step 5.Explain the Programme

The programme and its objectives need to be explained, in particular to the project staff who will be providing the data used by the programme and eventually will be involved in the resulting improvement initiatives. Once the programme has been approved by the Executive Sponsor, a series of short meetings should be arranged to explain the programme, its objectives and how it will work.

- Workshop for managers
- Workshop for project teams and developers
- Training for staff in measurement and analysis

As the programme proceeds, staff need feedback, incentives and recognition for their contribution. Results of the measurement programme and consequent proposals for process improvements should be discussed as openly as possible with staff before being implemented. As well as helping to ensure improvement are understood and supported, this also helps to demonstrate the impact and benefits being achieved by the programme. Questions which should be answered by the programme at this point include:

- do teams understand the purpose of the programme?
- do teams support the programme?
- do teams know what is expected of them?

Step 6.Devise Success Criteria

The measurement programme needs to be reviewed periodically, to ensure that it is functioning effectively and that it is generating information directly useful for assessing achievement against business goals:

- is the right information being generated at the right time?
- what is the quality of the information?
- have baselines and targets been set for each KPI?
- *have we got quantifiable information to aid decision making?*
- *has information been acted upon?*
- is it understood when to act on the information?
- has feedback been provided?

One of the key factors in a measurement programme's success will be the extent to which management see the quantifiable information as assisting decision making - equally, the extent to which this information has been acted upon to improve processes and provide tangible benefits will be an important indicator of the programme's value.

Step 7.Set baselines and targets for KPIs

The KPIs represent the information on which management will act and therefore will be linked directly to business goals. From the viewpoint of improvement, baselines and targets need to be set to provide the basis for answering the two most important questions:

Where are we? Where do we want to get to?

As the measurement programme proceeds, it will itself feedback to provide fuller information on the answers to both these questions. To assist this process, the programme should:

- identify when projects provide data and when data is assessed
- establish annual (or more frequent) baselines and targets, derived from quantifiable information
- assess performance against baselines and targets
- use data generated in first 3-6 months to set initial baselines
- base initial targets on achievements of known improvement programmes
- review and revise baselines and targets when measured information becomes available

Initial expected values (or ranges) of the KPIs can be established on the basis of available evidence, judgement and perception. This starts the iterative process of establish and assessing the current performance revealed by quantitative measurement.

Step 8.Develop a Feedback Mechanism

A feedback mechanism needs effective methods of presenting and reporting results. Figure 10 shows how project status can be reported in terms of achievement of targets such as cost and time and the extent to which project requirements have been met (e.g. as determined by project reviews or testing).

Figure 10. Project Status Reporting

	Status	*Cost*	*Time*	*Requirements*
Project A	Red	**X**	OK	OK
Project B	Green	OK	OK	OK
Project C	Amber	~	~	OK
Project D	Red	~	**X**	**X**
etc.				
Project X	Green	OK	OK	OK

An example of a more quantitative form of reporting is provided in Figure 11, for Development Capability. Various indicators of development capability are expressed on a graduated scale. The further the rating from the centre the more improvement is required. Distance from the centre can either represent the percentage by which the target has failed to be achieved or a graded rating from “acceptable” to “unacceptable”. Successive charts can show changes or trends in performance and the effects of process improvement initiatives.

Figure 11. Example 2 - Development Capability

The eight measures are:

- conformance
- resources
- tools and techniques
- hardware
- software
- effort
- timescales
- documentation

and can be plotted graphically as eight axes of a Kiviat diagram. This type of representation can also be a feature of process improvement approaches such as SPICE, where the desired process profile is compared with that determined by assessment of the development organisation's practice. The approach to a measurement programme described in this paper can be made consistent with the type of mechanism which can contribute to an organisation demonstrating capability by self-assessment or independent assessment against future SPICE or equivalent standards.

Process Maturity and Management

Process improvement models such as CMM, Bootstrap and SPICE define levels of organisation and maturity and associated generic "key process areas". Measurement is neither possible nor meaningful at the lowest levels (0,1) and have limited applicability at medium levels (2,3). They are essential at the highest levels (4,5). The models indicate the processes expected at each level and, hence, the measurements possible. Effective process management is possible from medium levels upwards.

Conclusions

This paper has described how a software measurement programme can be based on a small set of Key Process Indicators (KPIs) linked to management goals and simple project measurements and models for converting the resulting values to KPIs. This approach was developed for the METKIT training packages and Measurement Starter Kit, which have been purchased by over 100 organisations in Europe and elsewhere since its launch in 1993. It provides a simple but powerful means of obtaining and analysing software project measurements linked to business goals as an essential step to process improvement.

* * * * * * * * * *

References

1. METKIT Industrial Package. METKIT Consortium and Brameur Ltd, 1994.
2. A Starter Kit for Setting up a Measurement Programme, Brameur Ltd 1994

Practical Guidelines for Measurement-Based Process Improvement[4]

Lionel C. Briand, Christiane M. Differding, and H. Dieter Rombach[5]

Abstract
Despite significant progress in the last 15 years, implementing a successful measurement program for software development is still a challenging undertaking. Most problems are not of theoretical but of methodological or practical nature. In this article, we present lessons learned from experiences with goal-oriented measurement. We structure them into practical guidelines for efficient and useful software measurement aimed at process improvement in industry. Issues related to setting measurement goals, defining explicit measurement models, and implementing data collection procedures are addressed from a practical perspective. In addition, guidelines for using measurement in the context of process improvement are provided.
Keywords: software measurement, Goal Question Metric paradigm, process improvement

Introduction

Software measurement is widely recognized as an effective means to understand, monitor, control, predict, and improve software development and maintenance projects. However, effective software measurement requires that a great deal of information, models, and decisions be documented. Thus, it is a particularly difficult task for people who do not have extensive experience with software measurement. We provide here structured guidelines to address the issues most commonly encountered when planning and implementing a measurement program in the context of process improvement. This work is based in part on the authors' experience with measuring software development products and processes in the context of continuous improvement programs.
Measurement is introduced in software organizations to gain quantitative insight into the development processes and the developed products. This is important in order to understand better the development process, to identify problems and improvement opportunities. Measurement activities are commonly referred to as measurement programs. A measurement plan specifies the why, what, how, and who of a measurement program.
Goal-oriented measurement is the definition of a measurement program based on explicit and precisely defined goals that state how measurement will be used. In addition, explicit models have to be defined to support the derivation of questions and measures from the goals in a traceable and unambiguous manner. Goal-oriented measurement helps

- ensure adequacy, consistency, and completeness of the measurement plan.
- provide traceability between improvement goals, measurement goals, and measurement itself.
- stimulate a structured discussion and promote consensus about measurement and process improvement.

Our guidelines are defined in the framework of the GQM paradigm [BW84, BR88, Rom91, Bas93]. Nevertheless, it has to be noted that we do not always comply with the original GQM definitions and templates since we adapted them based on experience and projects' feedback.

---

[1] A complete version of this paper is published as Technical Report of the International Software Engineering Network (ISERN-96-05).

[5] L. Briand and D. Rombach ({briand, rombach}@iese.fhg.de) are with the Fraunhofer Institute for Experimental Software Engineering, Sauerwiesen 6, D-67661 Kaiserslautern, Germany. C. Differding (differdi@informatik.uni-kl.de) and D. Rombach are with the Software Engineering Group, Department of Computer Science, University of Kaiserslautern, D-67653 Kaiserslautern, Germany.

Goal-oriented measurement is performed through six major steps which are briefly described below. For more details, see [GHW95]. The process steps will be used as reference points throughout the paper so that the guidelines we provide can be mapped back into this measurement process.

Step 1: Characterize the environment. Identify relevant characteristics of the organization and of the project(s) to be measured. Typical questions are: What kind of product is being developed? What process is being used? What are the main problems encountered during projects?

Step 2: Identify measurement goals and develop measurement plans. Define the measurement goals based on the information gathered during Step 1. For each measurement goal derive the important attributes to be measured by involving project personnel and/or management. Document the definition of the measures and their underlying motivations in the measurement plan.

Step 3: Define data collection procedures. For all measures identified during the second step, data collection procedures have to be defined, i.e., how and when the data has to be collected and who will collect it. To optimize data collection procedures and limit data collection effort, the development process is a major element to take into account.

Step 4: Collect, analyze and interpret data. Collect project data, analyze them and interpret the analysis results with the help of project personnel and management.

Step 5: Perform post-mortem analysis and interpret data. Analyze the data further to identify, at the project level, the lessons learned from the entire project**.** In addition, project data are usually compared to past projects' data to identify specificities and explain differences.

Step 6: Package experience. Structure and store data analysis results, lessons learned, and related documents concerning the project and its measurement program in a reusable form.

Section 2 addresses the issues related to defining relevant measurement goals in an organization. The structure of GQM measurement plans, as we see them, is described in Section 3. Their implementation and all related practical issues are discussed in Section 4. Section 5 provides some insight into the interpretation of analysis results. Finally, Section 6 identifies typical measurement-based actions for improving the development process.

Definition of Measurement Goals

In this section, we introduce a modifed version of the GQM goal templates to guide the definition of measurement goals and discuss the main factors influencing their definitions. The section provides guidelines regarding Step 2 of the process for goal-oriented measurement defined above.

Applying GQM Templates to Define Measurement Goals

Practice has shown the importance of specifying a measurement goal precisely since the selection and definition of suitable and useful measures and models depends strongly on the clarity of these early decisions [BBC+96, BR88]. GQM provides templates for defining measurement goals in a precise way. This section describes the important aspects of these templates and provide examples. GQM templates structure a measurement goal based on five dimensions:

- The *object of study* defines the primary target of the study, i.e., the process or product that will be analyzed. Examples of objects are the entire development process, phases like system test, and documents like the design document, or the final project deliverable.
- The *purpose* of the study expresses why the object will be analyzed. Common purposes, in increasing order of difficulty, are:
    - *Characterization* aims at forming a snapshot of the current state/performance of the software development processes or products.
    - *Monitoring* aims at following the trends/evolution of the performance/state of some processes or products.
    - *Evaluation* aims at comparing and assessing the quality of products and the efficiency/effectiveness of processes.
    - *Prediction* aims at identifying relationships between various process and product factors and using these relationships to predict relevant external attributes [Fen91] of products and processes.
    - *Control* and *change* aim at identifying causal relationships that influence the state/performance of processes and products. *Control* consists in influencing the course of a project in order to alleviate risks. On the other hand, *Change* implies modifying the process from project to project in order to improve quality or productivity. *Change* requires usually a finer grain understanding of the phenomena under study than *control*.

- The *quality focus* states the particular attribute of the object of study that will be characterized, evaluated, predicted, monitored, controlled, or changed. Examples for quality focuses are cost, reliability, correctness, defect removal, changes, user friendliness, maintainability, etc.
- The *viewpoint* identifies the roles or positions of the people who are going to use the output of the measurement program, e.g., who interprets the data collected and uses the prediction models. These people are expected to provide strong input into the definition of the measurement program. Examples for *viewpoints* are project leader, developer, system tester, quality assurance manager, user, management, etc.
- The *context* of the study specifies the environment in which the study will be performed and correspondingly determines how generalizable the results will be. The information contained in the context is used to make environmental influential factors explicit, e.g., team structure and experience, application domain.

These five dimensions specify completely a measurement goal [BR88]. An example of a measurement goal using the GQM goal template is:

Analyze *the final product*
for the purpose of *characterization*
with respect to *reliability*
from the viewpoint of *the tester*
in the context of *Project X*

Every measurement goal can be expressed using this template. Goals should not cluster more than one purpose, quality focus, or viewpoint. Even though they may require similar data, this is likely to create confusion regarding which data are needed for which goal.

Factors Affecting the Definition of Measurement Goals

The definition of measurement goals is influenced by two categories of factors related to improvement goals and the development process in place. Software development organizations may have various kinds of improvement goals, e.g., reduce their cycle time and/or cost, improve the quality of their software, gain more control over their projects. From such *improvement goals*, one may derive *measurement goals* that help achieve these improvement goals, e.g., identify costly or errorprone actvities, identify the main sources of critical defects. In general, measurement goals may be derived from improvement goals in order to:

- provide relevant information to better manage projects
- provide relevant information to determine potential areas of improvement
- assess new techniques, methods, and standards quantitatively

Table 1 provides a structured overview of the impact of improvement goals and the development process on measurement goals. Rows contain the five goal dimensions. Columns contain aspects of the improvement goals and development process affecting the goal dimensions.
Knowledge concerning the development processes is needed in order to derive relevant measurement issues. Process descriptions include phases of development, the activities that are taking place during phases, the roles and positions involved in activities, and the development artifacts produced. Assessments based on some descriptive model of the process can help identify problems precisely and therefore help run a well focused measurement program. Such assessments can be performed through structured interviews, questionnaires, and defect causal analysis [BBK+94]. Indeed, they might point out issues to be investigated further through measurement. For example, do specification errors have costly consequences? Are most faults detected early? Is rework a substantial percentage of the development effort**?** [BDR96] contains a detailed discussion of this table.

Practical Constraints

This section illustrates constraints on starting a measurement program and establishing high-priority measurement goals.

Types of Goals

There are various environmental constraints which determine the types of goals which can realistically be achieved with measurement:
Resources

The scope of the measurement goal has to be adjusted to the resources dedicated to process improvement and measurement. One way to do so is to limit the *viewpoints* considered and the *context* of application of measurement.

Organization Maturity

The maturity of organizations has an impact on the definition of measurement goals. (Maturity is meant in the SEI Capability Maturity Model sense [PCC+93].) In cases where the practices and processes in place are unstable, characterization goals will provide less accurate results because variability will introduce uncertainty in characterization results. For example, developers will misclassify fault introduction phases because fault introduction phases mean different things to different people. However, it is important to note that data from unstable processes may be sufficiently reliable to partially or fully satisfy improvement goals.

State of Measurement and Process Modeling

Not any measurement goal can be achieved by any organization at any stage in their measurement program and process modeling activities. For example, it is often necessary to start with *characterization* or *monitoring* goals before *evaluation, prediction, control,* or *change* goals. An organization that does not understand how its resources are spent, what its most urgent problems are, and what the main causes of those problems are, should not assess new technologies. In order to construct a useful prediction model for process management, the organization's processes have to be understood from both a qualitative and quantitative point of view. If prediction goals are not achievable then control or change goals are out of reach since no relationships can be clearly identified.

	Factors	
Goal Dimensions	***Improvement goals***	***Development process***
Object of Study	Object of study should focus on products or processes that need to be better understood.	• Use process model to identify possible objects of study • Definitions of the Objects contained in process model can be used
Purpose	Purpose must be adapted to the level of understanding of problems in the organization.	Purpose must be adapted to: • Maturity of organization • Stability of the process • Control over process conformance
Quality Focus	Quality Focus should be consistent with the priorities of the corporate improvement program (market forces, company image, ...).	Quality Focus should address the most urgent weaknesses related to the process
Viewpoint	Depending on the improvement goals, one determines the activities the most in need of measurement. The personnel who performs the activities is the selected viewpoint. These activities are identified by specifying which are the most serious management and technical problems.	• From the roles involved in the development process, identify the viewpoints to consider • The descriptive process model contains a definition of the tasks associated with these viewpoints
Context	• Choose projects that are the most in need of improvement • Choose projects that are key to the success of the organization • Consider the resources dedicated to process improvement to determine the context	Determine the scope of measurement program by selecting a set of projects with: • similar process • similar application domain or focus on phases and activities in need for improvement

Table 1: Overview of factors influencing the dimensions of GQM goals

Number of Goals

In general, it is a good strategy to start with a small number of goals, gain experience, and then develop the measurement program further. The larger the number of measurement goals, the higher is the cost of measurement. This is especially true for the first goals of a program. It is important to demonstrate that measurement is useful to everybody in the organization, from both technical and managerial viewpoints. In other words, the measurement goals should address some of the issues raised by all categories of personnel at the project or organizational level. Everybody (high-level managers, project leaders, technical leaders, and developers) should feel they have something to gain in supporting such a measurement program.

Summary Table: Types of Constraints for Goal Setting
• Resources dedicated to process improvement • Depth of understanding of the current processes • Stability of the processes • Viewpoints (managers, developers,...) involved in the measurement program

Construction of a GQM Measurement Plan

GQM measurement plans contain the information that is needed to plan measurement and to perform data collection and analysis. This section explains the elements and the construction of GQM plans and refers to Step 2 of the measurement process: Identify goals and develop measurement plan.

Components of GQM Plans

A GQM plan consists of a goal and a set of questions, models, and measures. The plan defines precisely why the measures are defined and how they are going to be used. The questions identify the information required to achieve the goal and the measures define operationally the data to be collected to answer the questions. A model uses the data collected as input to generate the answers to the questions. The various concepts are briefly discussed in the following subsections.

Questions

The questions identify the information required to achieve the goal and the measures define operationally the data to be collected to answer the questions. A model uses the data collected as input to generate answers to questions. An example of a question would be “What is the quality of requirements documents?“ A quality evaluation model of requirement documents would be required to answer this question. Usually, GQM plans are composed of a large number of questions and [BR88] proposed categories of questions which can be used as guidelines.

Measures

Measures are operational definitions [JSK91] of attributes such as the quality focus and the factors that may affect it. Goals and questions may be defined without providing a specific operational model for attributes such as productivity or complexity. The next step is to provide operational definitions for those attributes so that they can be measured. Some attributes are actually based on several more elementary attributes, e.g., productivity, which is based on product size and effort. Therefore such attributes need to be operationalized through models that have as parameters more basic measures, e.g., Defect_Density- = Number of defects/LOC.
Defining a measure also includes defining its measurement scale and its value range. The scale is needed to guide the selection of analysis procedures once the data is collected. The range gives information on what data values are expected and may help detect abnormal values. For interval and ratio scale data, the measurement analyst has to specify the unit of measurement. For nominal scales and ordinal scales of limited range, the measurement analyst has to state precise semantics of all possible values. For example, assuming there is a measure capturing the tester’s experience, the scale could be ordinal and the range could be composed of the High, Medium and Low experience levels. As an example, the High, Medium, Low scores may be defined, respectively, as having developed functional test cases for more than five systems, at least one system, and never. Intervals or scores should be defined so that measurement results show variability across the scale. When data are collected through surveys and/or interviews, then their reliability should be studied carefully by assessing the measurement instruments**,** e.g., questionnaires, before the start of the measurement program.

Models

During the definition of GQM plans, different kinds of quantitative models have to be defined for the following reasons:

- GQM plans have to be operationalized. Therefore, the various abstract attributes of the artifacts being studied, e.g., maintainability, reusability of software components have to be defined in an operational way. We refer to such models as *descriptive* models. Building descriptive models is a matter of capturing expert's and practitioners' intuition into a quantitative model, e.g., define in quantitative terms what defect density is.
- The way quality or productivity comparisons and evaluations will be performed has to be defined precisely, i.e., how does an object or a set of objects compare to another object or population of objects with respect to a given attribute (i.e., the *quality focus*) or rather, to be precise, one or several of its measures. We refer to such models as *evaluation* models. For example, is a component overly complex or difficult to maintain based on its internal characteristics? Such decision functions can be built based on
    - expert opinion and captured decision algorithms that are based on intuition and experience.
    - the analysis of historical data related to actual decisions.
- The way predictions will be performed has to be defined precisely. Therefore, several questions must be considered:
    - What will the functional form of the models be? Should the models be linear/non-linear, univariate/multivariate, or take into account interactions between covariates?
    - What model building technique will be used? Is multiple regression analysis adequate?
    - What explanatory variables will be used to predict the dependent variable? For example, will system size, team experience, and application domain be sufficient to predict system cost?

  We refer to models describing these aspects as *predictive* models. Many techniques may be used to build such prediction models such as:
    - Regression analysis [BTH93, BMB94]
    - Inductive algorithms, e.g., classification trees, Optimized Set Reduction [BBH93]
    - Neural networks [KPM92]

Such models are usually of limited scope and are based on assumptions that are specific to the environment where they are defined. It is important to define descriptive, evaluation, and predictive models during the definition of the GQM plan since they will drive, to some extent, the definition of the measures to be collected and the definition of data collection procedures. For example, models may impose requirements on the type of measurement scale needed (e.g., complexity must be measured on an interval scale since a regression-based predictive model will be used) or on the reliability of the data collection (e.g., high reliability is required for a measure since it is expected to be one of the main predictors in many predictive models). [BDR96] contains a more detailed discussion of these models and how they are combined to be used for different measurement purposes.

Summary Table: Types of models
• Descriptive models operationalize attributes.
• Evaluation models are decision functions based on attributes.
• Predictive models predict external attributes of the object of study.

The Construction of GQM Plans

GQM plans tend to become large and complex because they include a great deal of information and interdependent concepts. Two kinds of documents provide support in constructing adequate GQM plans: abstraction sheets and descriptive process models.

Abstraction Sheets

GQM plans are constructed by defining and combining questions, measures and models based on the *viewpoints*' experience (see Section 2.1). The *viewpoint* does not need to see all the details of the GQM plan. The GQM plan is constructed by the measurement analyst based on the *viewpoint's* experience. To support the structured interaction of the measurement analyst with the *viewpoint*, a simplified view of GQM plans has been

designed [DHL96]. The documents are called GQM abstraction sheets and are used specifically for the purpose of facilitating interactions with *viewpoints*.
In order to capture the experience of the *viewpoints*, the GQM abstraction sheets are used as support documentation during interviews. Their components, referred to as *quadrants*, cover the issues that *viewpoints* need to address during interviews. Abstraction sheets may be viewed as structured guidelines to involve the *viewpoints* into the definition of the measurement plan. The GQM abstraction sheet completed in interviews is a major input when constructing the GQM plan since this is the main way of integrating the *viewpoints*' goals, experience, and feedback.
The suggested layout for the components of a GQM abstraction sheet is shown in Figure 1. In the following, the content of each quadrant is described:

- *Quality focus:* This quadrant is intended to capture the *viewpoints*' intuition about the *quality focus*, e.g., effectiveness, and help provide an operational definition for it.
- *Baseline hypothesis:* This quadrant specifies what is expected by the *viewpoints* with respect to the measures and models that define the *quality focus*. For example, if the *quality focus* measures the distribution of defects across classes of faults, the *baseline hypothesis* would specify the expected distribution of faults across classes. The values of this expected baseline can be based on data that have been collected during earlier projects or, if there are no relevant data, on the estimation and intuition of the *viewpoints*.
- *Variation factors:* This quadrant captures the factors that are believed by the *viewpoint* to have an impact on the *quality focus*.
- *Impact on baseline hypothesis*: The expected impact of the *variation factors* on the *quality focus* are captured here. Every *variation factor* must relate to the *quality focus*. The stated relationship between *variation factors* and *quality focus* must be testable. For example, the size of artifacts used as inputs by an activity could be considered a *variation factor* of the activity's effort. In this case, the impact on the *baseline hypothesis* could be stated as: "The larger the input artifact, the more costly the activity." This expected impact of the *variation factor* is the motivation for including the factor in the process or product definition category of the GQM plan.

Quality focus: What is module quality? e.g., # faults detected	**Variation factors:** What influences quality? e.g., module complexity
Baseline hypothesis: What is expected? e.g., < 5 faults per module	**Impact on baseline hypothesis:** How is the baseline influenced? e.g., highly complex module implies more faults

Figure 1: Structure of the abstraction sheet

A descriptive model for a quantitative attribute would require the definition of a measurement scale and precise semantics. For example, testing effort will be computed in person-days and will include the following activities: defining test cases, running test cases, checking test outputs, and writing test reports. Such activities would be precisely defined by a descriptive process model.
Once abstraction sheets have been completed, a first assessment of the size of the measurement program can be performed. Measurement analysts and users may decide to restrict the scope of the program, i.e., by restricting the number of viewpoints and the application context, in order to decrease the number of variation factors to be considered. In addition, factors judged as secondary may be left out.
In addition to being used as an instrument to support interviews during the definition of GQM plans, abstraction sheets may be used to show a simplified view of the GQM plan to project personnel. This will make any discussion of the GQM plan easier.

Using Descriptive Process Models

In the context of a measurement program, descriptive process models are needed for the following reasons:

- The definition of a measurement program and its data collection procedures requires knowledge of the process under study.

- Designing unintrusive measurement programs that fit into the actual process [BDT96, BBC+96] is a crucial requirement.
- The data collected will not be interpretable and amenable to process improvement if analyzed in a vacuum, without a good qualitative understanding of the process [BBC+96, BBK+94].
- Discussions, decisions about changes, and communication of improvement decisions in an organization will require some widely-accepted model of the process under study.

The information items of descriptive process models relevant to defining a GQM plan can be classified into at least three categories:

- Definitions of phases and activities, and the data/control flows that relate them.
- Characterization of produced artifacts and their various states (e.g., under the form of a state-transition diagram) during the development process.
- Positions and associated roles in the organization, i.e., responsibilities with respect to activities and produced artifacts.

Summary Table: Components of a GQM plan
• Goal(s) • One Abstraction sheet per goal • Questions • Models: descriptive, evaluation, predictive • Measures

Implementation of GQM Plans

Based upon GQM plans, specific data collection procedures are designed in a way so that reliable data can be collected in the environment and process under study. This section provides guidelines for their definition. In addition, practical issues, which are crucial for the successful implementation of measurement, are discussed. This section refers to Step 3 and Step 4 of the measurement process.

Defining Data Collection Procedures

After measures have been defined for each GQM goal, they have to be mapped to precise data collection procedures that provide the required level of data reliability at low cost. When developing the data collection procedures, decisions concerning the point in time, the responsible person, and the best means for data collection have to be made. The descriptive process model provides an important input for these decisions. [BDR96] contains a more detailed discussion of these aspects.

When to collect the data

According to the measurement purpose and data collected, three main types of strategies can be adopted for data collection: periodically, at the beginning/end of activities and/or phases, and when an artifact has reached a certain state.

These strategies support, respectively, three categories of application of measurement:~~:~~ Monitoring and control of software development projects, process improvement (within or between projects), support of quality assurance activities. Table 2 shows for each of these strategies the measurement purpose they support, examples of typical data to be collected and the main inputs needed from the descriptive process model to design collection procedures.

Who Collects The Data

Another question is to determine who can and/or should collect the data, and whether a tool can automate the data collection. If the answer is no, then subjectivity in measurement cannot be avoided. In this case, several criteria can be adopted to determine the right person(s) to collect the data: expertise, bias, access, cost, availability, and motivation.

There~~y~~ are three main categories of measurement instruments: tools, questionnaires, and structured interviews. The decision about which instrument to use depends on the information collected. Tools can be used for

objective artifact measures (e.g., LOC), questionnaires and structured interviews for process measures (e.g., effort spent on an activity) and subjective measures (e.g., understandability of the requirements.)

Collection strategy	Measurement purpose	Examples	Inputs needed
Periodically	monitoring and control of projects	% of modules tested cumulative effort over time	Level of granularity of updates, e.g., weekly, monthly.
Beginning/End of activities	process improvement: Identification of inefficient and/or ineffective activities	defect detection rates and cost of testing activities	descriptive model of activities and artifacts used or produced by processes
Artifact States	quality assurance support: Identification of defect-prone or costly components	effort spent on inspection, what is the observed quality?	state-transitions diagrams of products

Table 2: Strategies for designing data collection procedures

How to Collect Data

There are several issues of importance for the acceptance of questionnaires. ~~It is important that f~~Forms filled out by project personnel should be designed so that each person has to fill out only one specific form at a time. For a better acceptance of the data collection procedures by personnel, the forms should be adapted to the terminology, procedures, and tools (e.g., SEEs, CASE) used in the project.
Filling out the forms ~~out~~ should be perceived as a natural part of the various activities, and should not be considered as an overhead by the management or personnel.
Table 3 summarizes the most important decision criteria concerning data collection procedures.

Type of decision	Decision criteria
When to collect the data	Application of measurement, i.e., monitoring, prediction, control, quality assurance.
Who collects the data	People's expertise, bias, data access, cost, availability, motivation
How to collect the data	Tools available, Procedures and Tools used in the project

Table 3: Decisions concerning data collection procedures

Practical Issues

An important principle of the GQM approach is that the project personnel who collects and uses the data participates actively in both the definition and the interpretation of the data. Thus they realize that the collected data is used to address their own needs and are motivated to provide reliable data. The participation of project personnel should cover the following activities:

- *Goal setting:* The measurement goals should concern developers as well as project and quality assurance management, so that the different project viewpoints are represented by the measurement program. This will increase the chances for acceptance because it serves the interests of all parties involved.
- *Measurement planning:* Planning, i.e., the definition of questions, models, and measures, requires the participation of project personnel. Thus, project personnel and management will be involved in all important decisions about measurement. This increases chances of acceptance because it will ensure that the measurement program is well-suited.
- *Data collection forms and procedures:* The data collectors should be involved in testing and reviewing the forms. Pretesting of the forms will provide evidence of the reliability of the data collected or the lack thereof.
- *Interpretation of data:* Despite the fact that measurement specialists have to analyze the data, sometimes using sophisticated statistical techniques, the interpretation of the results must be performed in close collaboration with the *viewpoints* and, possibly, the people who collected the data. The results of the data analysis performed by the measurement specialists are presented to and discussed with the *viewpoints* and, eventually, the data collectors in feedback sessions.

The project personnel involved in measurement must be trained in several topics in order to ensure wide acceptance for measurement and to get reliable data. The main topics to address by training are the following:

- The purposes of the measurement activities
- Fundamentals of the GQM approach
- Relevant issues concerning reliability of the collected data
- The data collection tools, for a more efficient and reliable tool usage

A discussion of commitment, training, and tool issues can be found in [BDR96].

Data Quality Assurance Procedures

When the data has been collected, it has to go through a quality assurance process before it can be stored or analyzed. The quality assurance process addresses the following issues:

- There may be data collection forms with missing data.
- Data collection forms may contain outliers to be checked or values that are out of range.
- Various dependencies between data collection forms and developed artifacts have to be checked for consistency.

Analysis, Presentation, and Discussion of Measurement Data

This section discusses the types of data analyses that may be relevant in the context of software measurement. Furthermore, recommended strategies for the dissemination, and interpretation of analysis results are discussed. The activities described in this section are part of Step 4 in the goal-oriented measurement process.

Comparison of Quality Focus Data with Baseline Hypotheses

The data collected can be used to build quantitative baselines for the development projects of the organization. It is usually interesting to compare actual baselines to the expected ones (i.e., baseline hypotheses as defined in abstraction sheets - see Section 3.2.1). This will allow the measurement analysts to:

- Explain these differences and determine whether they are symptomatic of a problem.
- Trigger discussions with developers, project leaders, and management.
- Show the usefulness of measurement by identifying departures from expectations.

It should be noted that the quantitative baselines and their comparison to the baseline hypotheses are computed based on the various models defined in the GQM plan (i.e., descriptive and evaluation models). For example, if one question asks about the distribution of effort across phases, the collected data are aggregated according to the descriptive model for effort distribution. This allows for the computation of the actual distribution of effort across phases. Then this quantitative baseline may be compared to the expected one (i.e., baseline hypothesis) through inference testing by comparing distributions and assessing the significance of their differences.
Significant differences between the baseline hypotheses and the actual data lead to issues that should be addressed in feedback sessions. Moreover, they are likely to trigger further investigation of the data in search for factors that explain the differences. For example, if the testing phase detects more defects than expected, the analyst would look at the quality of the documents (e.g., the documents may be of poor quality) and look at the testing technique (e.g., it may be more effective than usual). It should be noted that such an analysis of the baselines is a required component of the preparation for feedback sessions. Feedback sessions will help select the most probable explanations among plausible alternatives.

Variation Factors: Validation of the Variation Hypotheses

Depending on the *purpose* of the GQM goal, the following strategies are applied:

- For prediction purposes, the *variation hypotheses* are tested by answering the following question: Did the variation factors have the expected impact on the *quality focus*? If the expected impact cannot be verified, then excluding the variation factor from the data collection should be carefully considered. Otherwise, the identified relationships may be used to build new or more reliable models for project management, quality assurance, etc.
- For control and change purposes**,** assuming that the variation factors have already shown to be of some impact, the analysis concentrates on determining whether or not this impact is due to a causal relationship between the quality focus and the variation factors.

It should be noted that *variation factors* are not relevant in the case of characterization since this purpose focuses exclusively on providing a snapshot of the development processes and products, e.g., distributions of effort across phases or components across complexity levels.

Objectives of Feedback Sessions

The major objective of the feedback sessions is to interpret the data analysis results with the help of the *viewpoints* and the additional project personnel who has the necessary expertise. Therefore, the results are presented to the session participants and possible interpretations are discussed. The presentation and discussion is structured according to the stated GQM goals. Improvement possibilities concerning the development process or changes of the project plan may be considered by the participants. Moreover, the measurement program may be evaluated. If participants are not able to use the data, this may be explained in different ways:

- The results are not presented in an adequate form to the participants.
- The data may not fit the stated measurement goal, i.e., the defined measures do not adequately capture the attribute that one purports to measure.
- There may be some relevant information missing, i.e., some extraneous factors are not measured.

During the initial phases of a measurement program, these issues have to be considered carefully, because they ensure the completeness, consistency, and reliability of a measurement program. After a few projects, the measurement program should stabilize.
Subsequently to the feedback sessions, one should refine the data analysis, and, if necessary, the GQM plan and the collection procedures, based on newly acquired insights.

Summary Table: Objectives of Feedback Sessions
• Interpret trends identified by the data analysis
• Take corrective actions concerning the project, process, or measurement program
• Assess and refine the measurement plan

Organization of Feedback Sessions

Once the data collection has started, feedback sessions should be held periodically, e.g., intervals between sessions should be a matter of weeks. Their preparation consists of the following activities:

- Data analysis.
- Layout of results in comprehensible and intuitive ways.
- Identification of alternative interpretations.

The participants of the feedback sessions are the *viewpoints* of the GQM goals and the people who collected data. Both groups are important for the interpretation of the data and are likely to be affected by process and data collection changes that may be decided during the feedback sessions [GHW95].
The presentation material should be structured according to the GQM plans and contain at least all the issues identified by the analysis. The material should be distributed to the participants well before the feedback session so that they have a chance to look at the results beforehand.

Summary Table: Organization of Feedback Sessions
• They are held periodically
• Participants are data collectors, viewpoints, and measurement analysts
• Presentation material should be distributed well in advance

Interpretation of Results

The analysis results are interpreted by the *viewpoints* and, in some cases, the data collectors. *Viewpoints* will know how to use the results according to their objectives, and the data collectors know how well the data they provided were actually collected and whether they are suited to the objectives of the *viewpoints*. For example, the *viewpoint* may draw false conclusions from the small number of failures being reported, if the data collectors do not object that not every failure identified during test has been reported due to time pressure.
The *viewpoints* (and only them) can draw conclusions from the results that are highly dependent on the context of the measurement program and therefore more likely to be accurate. The underlying rationale leading to conclusions and all related explanations must be documented. This is necessary in order for those conclusions to be questioned and refined later on if inconsistent or complementary conclusions are drawn during subsequent feedback sessions.
The interpretation of the data should lead the identification of weaknesses of the processes in place and the discussions of possible improvement strategies.

Establishing a Process Improvement Action Plan

Our goal in this section is to provide a structured overview of measurement-based improvement opportunities and how to proceed with them. In the context of a goal-driven measurement program, lessons learned based on a thorough data analysis and interpretation lead to various opportunities for improvement. A non exhaustive list of typical recurring opportunities is provided below:

- Identification of unsuitable or low quality development artifacts
- Identification of error-prone and/or inefficient activities
- Interfacing problems between phases
- Management problems

The identification of improvement opportunities is based on existing descriptive process models and on a careful analysis of the distribution of effort and defects across phases, activities, and artifacts. In general, one should look at the following aspects:

- differences in proportion between categories of defects according to their type, origin, cause, etc.
- associations between defect categories and
  - phases/activities and life cycle products where introduced
  - phases/activities where detected
  - various products' parts, e.g., subsystems
- activities' and phases' relative effort and duration

Once problems have been clearly identified, the search for sound and economically viable solutions starts. However, new technologies and methods should be introduced with care in an organization. Any method, technique, and language should always be carefully evaluated before using it across the organization. Different types of empirical investigations may be used. The two main ones can be briefly and informally described as follows:

- *Case Studies:* One or a small number of pilot projects are monitored. The new technology is introduced on all pilot projects without any control on influencing factors. There is usually no "control" project where the new technology is not used and against which results can be compared. Results are interpreted by relying heavily on interviews and a careful qualitative analysis of the process. When data are collected, which is recommended, comparisons to the measurement-based baseline may be performed.
- *Controlled experiments*: The size of the sample (usually individuals) under study allows for the derivation of statistically significant results. The new technology is introduced on a part of the sample, the other part being used for comparison. These parts are selected randomly. The factors influencing the impact of the new technology are largely controlled for.

The descriptions above are a rough but relatively representative generalization. These two types of investigation represent the extreme points of a range of empirical research designs. Many intermediary strategies exist and may be better suited, e.g., quasi-experimental designs [JSK91].The two types of investigation have different drawbacks, strengths, and therefore purposes. We will briefly discuss them in the following paragraphs:

Case studies:

- *Strengths:* low cost, can be easily performed in a real field setting, useful to identify new issues to be investigated, suited to understand the why and how of phenomena.
- *Weaknesses:* no statistically significant results can be obtained, many threats to the validity of the conclusions that can be drawn, more difficult to perform well (e.g., concerning data analysis) and requires high application domain expertise, difficult to ensure that the results are generalizable.

Controlled Experiments:

- *Strengths:* statistically significant results, causal relationship may be demonstrated, effects of new technology may be more precisely estimated.

- *Weaknesses:* high cost, difficult to perform in field setting, only useful for (dis)confirming well stated hypotheses and theories.

One effective strategy is to combine the use of controlled experiments during training exercises and case studies on pilot projects. Because these two investigation strategies have complementary weaknesses and strengths, if consistent results are obtained, each investigation reinforces the other's results.

Conclusion

Setting up a successful measurement program for process improvement is a necessity but challenging undertaking. The reasons are multiple. Measurement needs to be performed from various points of views, encompasses numerous attributes, models, and interdependencies between them. Furthermore, many psychological issues have to be addressed to increase chances of success.
For this reason, goal-oriented measurement combined with explicit modeling (e.g., process, quality, etc.) can greatly help structure and provide rigor to the measurement plan. This in turn allows for completeness and consistency analysis of the plan. In addition, communication among the measurement program participants and users is improved, because supported by clear and explicit documentation.
In this paper, we have provided practical guidelines to all the steps required to address the issues mentioned above and to increase the chances of measurement to lead to actual process improvement. Additional guidelines concerning the implementation of the measurement plan (collection, analysis, interpretation) are given within the context of the GQM paradigm.
Future work includes formalizing better the structure and content of the measurement plan so that better automated support can be provided. Thus, the complexity will be easier to cope with for the measurement analysts and improved guidelines will be available for data collectors.

Acknowledgments We thank Frank Bomarius, Khaled El Emam, Christopher Lott, Sandro Morasca, Dietmar Pfahl, Carsten Tautz, and Isabella Wieczorek for their comments on the paper.

References

[Bas93] V. "Applying the Goal/Question/Metric Paradigm in the Experience Factory." Presented at the 10th Annual CSR Workshop in Amsterdam, October 1993, to appear in a book entitled *Software Quality Assurance: A Worldwide Perspective*, by Chapman and Hall.

[BBC+96] V. Basili, L. Briand, S. Condon, Y. Kim, W. Melo, and J. Valett. "Understanding and Predicting the Process of Software Maintenance Releases." In Proceedings of *the 18th IEEE International Conference on Software Engineering*, pages 464-474, Berlin, Germany, March 1996.

[BR88] V. Basili and H. D. Rombach, "The TAME Project: Towards Improvement- Oriented Software Environments", *IEEE Transactions on Software Engineering*, 14 (6), pages 758-773, June, 1988.

[BW84] V. Basili and D. Weiss, "A methodology for collecting valid Software Engineering Data", *IEEE Transactions on Software Engineering*, 10 (6), pages 728-738, November 1984.

[BBH93] L. Briand, V. Basili and C. Hetmanski. "Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components," *IEEE Trans. Software Eng.*, SE-19 (11):1028-1044.

[BBK+94] L. Briand, V. Basili, Y.M. Kim, and D. R. Squier. "A Change Analysis Process to Characterize Software Maintenance Projects." In Proceedings of the *International Conference on Software Maintenance*, pages 38-49, Victoria, Canada,1994.

[BDR96] L. Briand, C. Differding, and D. Rombach. "Practical Guidelines for Goal-oriented Measurment". Technical report of the International Software Engineering Network (ISERN-96-05), Kaiserslautern, Germany, 1996.

[BMB94] L. Briand, S. Morasca, and V. Basili, "Defining and Validating High-Level Design Metrics", CS-TR-3301, Version 2, University of Maryland, College Park, MD, 20742, April 1994. Submitted for publication.

[BTH93] L. Briand, W. Thomas, and C. Hetmanski, "Modeling and Managing Risk Early in Software Development", in Proceedings of *the 15th International Conference on Software Engineering*, pages 55-65, Maryland, May 1993.

[BDT96] A. Broeckers, C. Differding, and G. Threin. "The Role of Software Process modeling in Planning Industrial Measurement Programs." In Proceedings of *the Third International IEEE Software Metrics Symposium*, pages 31-40, Berlin, March 1996.

[DHL96] C. Differding, B. Hoisl, and C. Lott. "Technology Package for the Goal Question Metric Paradigm." Internal Report 281-96, Department of Computer Science, University of Kaiserslautern, 67653, Kaiserslautern, Germany, April 1996.

[Fen91] N. Fenton. "Software Metrics: A rigorous approach.", Chapman & Hall, 1991, London.

[GHW95] C. Gresse, B. Hoisl, and J. Wuest. "A Process Model for Planning GQM-based Measurement." Technical Report STTI-95-04-E, Software Technology Transfer Initiative Kaiserslautern, University of Kaiserslautern, Germany, October 1995.

[JSK91] C. M. Judd, E. R. Smith, and L. H. Kidder. Research*, "Methods in Social Relations*." Harcourt Brace Jovanovich College Publishers, 1991.

[KPM92] T.M. Khoshgoftaar, A.S. Panday, and H.B. More. "A Neural Network Approach for Predicting Software Development Faults." In Proceedings of *the Third International IEEE Symposium on Software Reliability Engineering*, pages 83-89, North Carolina, October 1992.

[PCC+91] M. C. Paulk, W. Curtis, M. B. Chrissis, and C. V. Weber. "Capability Maturity Model, Version 1.1."*IEEE Software*, 10(4), pages 18-27, July 1993.

[Rom91] H. D. Rombach. "Practical Benefits of Goal-Oriented Measurement." In N. Fenton and B. Littlewood, editors, Software Reliability and Metrics, pages 217-235. Elsevier Applied Science, London, 1991.

SP 96 Conference - Brighton - December 1996 - Session 2: ISCN 96

Process Improvement in SMEs: the Onion Experience in Internet Service Providing

G. Bazzana, E. Fagnoni, M. Piotti, G. Rumi
ONION Communications, Technologies, Consulting
Via L. Gussalli, 11 - 25131 BRESCIA (Italy)
E-mail: info@onion.it - Web: http://net.onion.it/

Abstract

This paper presents the outcomes of a process improvement program, called PI^3 (Process Improvement In Internet service providing), run at ONION.

It covers the following aspects:

- the status of software engineering practices at the beginning of the improvement program, in terms of ONION development activities and weak/ strong process areas;
- the improvement plan defined to raise the maturity level and, above all, the development/ maintenance capabilities of the software producing unit;
- the steps in which the improvement program was organized, with emphasis on Testing and Configuration Management activities;
- innovative means for handling the Quality Management System with the support of a company Intranet;
- results achieved and lessons learnt.

Keywords

Software Process Improvement, Testing, Configuration Management, QMS, ISO 9000, Internet, WWW

Business Motivation

Onion Company Profile

Onion is a privately owned company based in northern Italy, offering advanced IT services; the company is specialised in the fields of Communications, Technologies and Consulting, always trying to keep a strong link between technology and business goals.
Onion is pursuing the following business areas:

- COMMUNICATIONS: Distributed Computing and Networking applications, Telematic Services, Internet/ Intranet Service Providing, Internet Access Providing
- TECHNOLOGIES: Security Management, Innovative Multimedia Applications, Information Technology Transfer, Information System Rightsizing
- CONSULTING: Business Process Re-engineering, ISO 9000 Quality Management Systems, Food and Drug Administration (FDA) Computer System Validation, Software Process Improvement.

More details can be found at the World-Wide-Web: http://net.onion.it/

Business Needs

Though being a young SME, Onion is very committed in strengthening its business capabilities through software process improvement.
Onion is intensively working in a technology environment which is evolving very rapidly. Nowadays, companies are more and more reliant on information which straddle national boundaries. Multinational corporations need their communication and information exchange capabilities to function efficiently in a global business environment.
To this aim, recently two forces of technological change have created a shock wave through the communications and computing industries and shaped the blueprint for the Information Highway.
First of all, in telecommunications, transmission is evolving from copper wire to fibre optic cable, along with a new generation of telecommunication switches and embedded software. The virtually limitless capacity of optical fibre has substantially eliminated capacity or bandwidth as a constraint. Moving the large amount of information required for sound, videos or images has become increasingly rapid and cost-effective.
The second technological breakthrough is the availability to convert text, sound, images, video and other content into a common digitised format. This fact makes it possible to connect all communication systems into a single vast network.
Internet is today the best known electronic network and, even if a lot of work still remains to be done, it represents how the new communication technologies have become more accessible. Internet already offers, albeit in embryonic form, most of the services and technologies that should make a substantial step ahead to our quality of life: you can make a telephone call, watch a video, listen to an audio, shop, learn and, of course, communicate. Many are the benefits that derive from this situation and the combination of abundant, low-cost information and its instant availability through a fast, efficient and ubiquitous electronic network gives companies the power to make economical use of the resources they need
This “IT revolution” is fundamentally affecting also the software development paradigms and the key technical strengths for competition.

The experiment and the baseline project

At Onion software development is a key factor for communications and technologies services/products; software related activities can be classified into the following three classes:

- software development for turn-key IT solutions: in this case software development follows a traditional waterfall life cycle with usage of C++ and Visual Basic as development languages;
- service providing on Internet (e.g.: Web server information publishing, support to customers' operations, set-up of company Intranets, support to order processing and inventory management, etc.): in this case software is "embedded" within the provided service and is developed with innovative languages like Perl, VRML, Java, etc.;
- development of multimedia applications: in this case software development cannot follow a standard waterfall model, but has to face with fast prototyping, Rapid Application Development (RAD) and integration of software with multimedia assets.

The Process Improvement program described in this paper focused on the second application domain, in which a typical project is characterized by the following phases:

- definition of service requirements with the customer;
- collection of assets to be included in the service;
- definition of the home page for the service;
- definition of search keywords;
- set-up of the service structure;
- development of prototype;
- testing of prototype;
- review of the service prototype with the customer;
- completion of service development;
- testing;
- fixing of failures found;
- acceptance with the customer;
- insertion of service in production environment;
- link of the service with most known searchers.

The Process Improvement Experiment

The starting scenario

Findings of a self-assessment

Onion conducted a software process self-assessment resulting in a maturity not aligned with the company strategies. Apart from the numeric grading, the assessment was very important in raising the consciousness about process improvement needs and in singling out key process areas that should be addressed first.

In the following the situation at the start of the improvement project is summarised from a technical, business, organisational, cultural point of view.

- Technical issues
  The self-assessment, combined with a portfolio analysis of business needs, brought to the identification of the following areas for improvement, with reference to SEI CMM Level 2 Key Process Areas (KPAs), in decreasing order of impact onto business goals:
  a) testing: testing was conducted by the developers, without adoption of any consolidated technique and with insufficient focus on users' and service features; hence, testing effectiveness had room for improvement;

b) software configuration management: there was provision neither for versioning, nor for change management; this caused a low productivity, a high degree of regression testing and an insufficient re-use of objects; such issues applied to software code, technical deliverables and also to published assets (pictures, photos, films, forms, etc.).

Strong key process areas at the start of the improvement program included Quality Assurance, with the presence of a Quality Manager committed to the enforcement of good engineering practices. Also requirements management was felt as satisfactory, especially in Rapid Application Development, where the life-cycle involves prototypes discussed with customers quite early and frequently. Project management also showed good foundations.

- Business issues
  From a business point of view, PI^3 assumed as pilot projects two of the most important developments undergoing in the company, in order to deploy the improvements as soon as possible in the core business areas, to take immediate advantage of the expected benefits. Concerning company capabilities, positive results from Process Improvement were expected in product reliability and development productivity/ timeliness; such issues were tracked by a set of quantitative indicators.
- Organisational issues
  Organisational issues at the beginning of the experiment were marked by a focus on technology and on people-driven processes. Hence the processes might be defined as “ad hoc” or “chaotic” with focus on short term goals and several deficiencies in medium-long term issues. The assessment also highlighted the organisational strengths that would have constituted the basis for process improvement, in particular: attention paid to training; adoption of a rough measurement system to track projects; consciousness of need for improvement and positive attitude towards it; competent and creative people, with a good mix of technical, managerial and commercial profiles; state-of-the-art technology.
- Cultural issues
  The willingness to improve of the whole development structure brought to the deployment of improvement actions in a positive framework, without resistance from the staff.

The Improvement Plan

After the assessment, an action plan was devised in order to increase the software development/ maintenance capabilities of the software producing unit.
The bulk of the improvement was planned to cover up to December 1997, accompanying the company from the incubation period, through take-off and growth consolidation. The plan has three main steps covering two years of elapsed time:

- short term improvement efforts (from June 1995 to December 1995)

  It was decided to strengthen training issues by means of the definition of a training plan (customised for various professional skills) and to enforce project management;

- medium term improvement efforts (from January 1996 to December 1996)

  This phase has the goal to address configuration management and testing issues that both require significant investments in technology and effort to be set-up and deployed;

- long term improvement efforts (from September 1996 to December 1997)

  This phase will focus on the drafting of the documents constituting the Quality Management System (QMS) of the company, allowing for ISO 9000 registration.

As a consequence of the assessment, it was decided to focus on Process Improvement actions characterised by the following characteristics: highest pay-off; relevance for all the business lines of the company; direct applicability and pragmatic feasibility in the medium term.

The key process areas exhibiting such characteristics were identified as being the following: configuration management (CM) and testing; for both of them, the PI^3 project looked at both the definition of rules and the alignment of projects.

Approach with respect to business needs

Technical implementation of the experiment

The phased work-plan of the project was defined taking into account the following strategy:

- the Improvement Program shall be based on top of running pilot projects;
- to handle the average length of projects and to track the trends of results, the application of the Plan-Do-Check-Act (PDCA) scheme was planned on two projects~~at least twice~~;
- each work-package of the PI^3 Program was defined as a major sub-division of the project, ending with a verifiable end point;
- project management was clearly identified also in terms of resources;
- sufficient room was devoted to training and dissemination activities;
- sufficient room~~s~~ was also devoted to quantitative measurement of results (in particular: effectiveness analysis and quantitative evaluation of Return On Investment - ROI).

In accordance with the goals of the overall improvement plan, the required status of software processes at the end of the improvement program can ~~shall~~ be summarised as follows:

- definition of processes and adoption of tools for the KPAs of testing and configuration management;
- adoption of the defined practices and of the tools in the daily routine work of projects;
- alignment of the staff to the defined methods, practices and tools;
- increased maturity of the measurement system, tracking the most relevant indicators for the business of the company;
- beginning of formalisation of experiences gained by means of standard operating procedures constituting an initial kernel of the company QMS.

The maturity level of the company at the end of the Improvement Program was planned to be not too far from level 2; this would be a big success for the organisation because in this case we would have achieved a stable process with a repeatable management control level, by initiating rigorous project management of commitments, costs, schedules and changes.
In particular the improvement program was planned to bring close to level 2 for the KPAs directly affected by the improvement actions. In particular we expected significant improvements in the following practices: Life Cycle Functions: testing; Supporting Functions: configuration management; Process Related Functions: process control; Technology: tools for configuration management and testing. Moreover, the Improvement Program was planned also to affect positively higher maturity level key practices, likewise: process definition and process measurement.
What is felt as most important in any case is not the fact of reaching a full level 2 for all practices but rather to have full alignment between the defined practices and the daily routine work within the projects; in this case in fact it will be possible to improve company capabilities by means of a bottom-up continuous improvement suggested and enforced by the whole staff and not just top-down driven by the management.

The improvement steps

The steps in which the improvement program has been organised can be summarised as follows:

1. Evaluation of the state of the art methods and tools
2. Procurement of the selected technology
3. Training on technology and underpinning methods
4. Definition of rules on how to apply the selected methods and tools to the pilot projects
5. Definition of quantitative measures to track the effectiveness of the improvement program
6. Application of the selected methods and tools to the pilot projects
7. Collection and analysis of quantitative data from the program experiment
8. Analysis of Return On Investment
9. Transfer of the lessons learnt to the whole staff
10. Transfer of the results into the standards operating procedures

Particularly important was the initial technology survey, that brought to:

- the evaluation and procurement of CM tooling (through: selection of candidate tool-kits, design of a checklist containing the most important aspects to be evaluated, comparative evaluation using the checklist, procurement);
- the evaluation and procurement of testing tooling (through the same approach described for CM);
- the drafting of a final report summarising the tools evaluated and giving a rationale for the procurement decision.

The changes that were made to the technical environment in terms of equipment, tools or other software introduced specifically for the process improvement activity are summarised in the following:

- adoption of a WWW Workbench including advanced authoring and testing features, as well as basic Configuration Management features;
- adoption of a WWW Test Environment covering almost all the needs that were defined at the beginning of the technology survey;
- adoption of a CM environment particularly suited for document and asset management, oriented also towards ISO 9000 document and data control rules;
- set-up of a WWW environment for the management of the Quality Management System.

In the next paragraphs technical details are reported of the most substantial improvement efforts made.

Improvement actions deployed

The Web Site Testing Procedure

Testing improvements

Concerning testing, improvement actions included, among others: test design methods (reference documents, methods for extracting test cases, etc.), practices for unit test, practices for integration test, practices for system/ acceptance test, methods for problem notification and tracking, test reporting.

This resulted in much more detailed testing activities than before, introducing test design and reporting rules and clearly identifying the testing steps among which the following were reckoned as particularly important:

- testing for service content and language;
- testing for hyper-textual links;
- testing for HW/software compatibility;
- testing for usability;
- testing for efficiency;
- regression testing.

Test classes and levels

The following test levels were defined depending on the application domain, either Programs (classic software development) or Web Sites.

Programs	*Web Sites*
Module Testing	Syntactic Testing
Integration Testing	Security Testing
System Test	Service testing

Tab. 1 - Test levels for various classes

While not detailing here the meaning of the testing levels associated to Programs (for references, see [BACH] or [BEIZ]), a few words are needed for test levels of Web Sites.
Syntactic tests have the goal to check the basic correctness of Web Sites, from a syntactic point a view, a structural point of view (in particular referring to "link resolution" aspects) and a performance point of view (in particular looking at the number and size of pictures).
Security tests have the goal to validate the security mechanisms and security enforcing functions, with particular emphasis on the reserved and restricted areas; when looking for security of critical systems, the usage of ITSEC [CEC1] and ITSEM [CEC2] guidelines can be extremely valuable.
Service tests have the goal to validate the resulting service from an user's point of view, thus adopting a black-box strategy without any assumption on the underlying architecture and implementation choices.

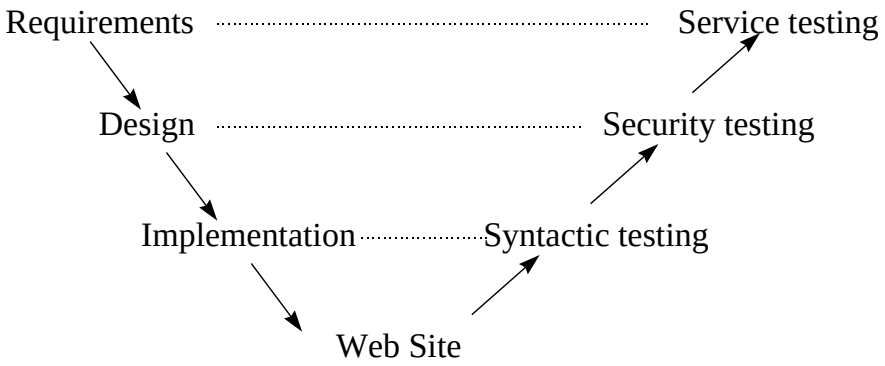


Fig. 1 - The V-cycle applied to WWW development

The Web Site check-list

In addition, a standard check-list was devised for all Web Sites, to be applied both for acceptance purposes and for regression testing activities. Such test-list covers aspects likewise:

- stylistic problems (spelling errors, particular tags, use of obsolete mark-up, particular content-free expression, empty container elements, etc.),
- lexical problems (use of character sets, formatting-related problems, using white spaces around element tags, etc.),
- syntax problems (illegal elements, illegal attributes, unclose container elements, malformed URLs and attribute values, etc.),

- image related problems (bandwidth consumption, images syntax, etc.),
- document structure problems (both in tables and in forms),
- portability problems (accessibility by various browsers and platforms, mark-up inside comments, use of single quotation marks for attribute value, use of specific mark-up not supported by all browsers, liberal usage of file naming, etc.),
- structural integrity problems (no index file for a directory, dead links, limbo pages, etc.),
- security problems.

The Testing Procedure

The adoption of supporting tools [IMAG] [BOWE] allowed to set-up a test factory running almost automatically the following sequence of test classes:

- check the document for spelling errors;
- perform an analysis of the images;
- test the document structure;
- look at image syntax;
- examine table structure;
- verify that all hyper-links are valid;
- examine form structure;
- analyze command hierarchy.

For more details on the Onion approach to testing in the Internet domain, the interested reader is referred to [BAZZ].

O-O based Configuration Management

Configuration Management improvements

Concerning Configuration Management, issues addressed included, among others: process management (life-cycles for various objects, user roles, triggers, security control, etc.), release management (versioning, object control, dependency management, build management, bill-of-materials, variants and parallel releases, etc.), change management (status handling, report handling, etc.) [MARK].

The following detailed activities were fulfilled:

- definition of rules for applying methods and tools within the pilot projects;
- application within pilot projects;
- derivation of guidelines for company-wide configuration management, for inclusion within a QMS standard operating procedure.

Process improvement in Configuration Management resulted in a more severe distinction between the development environment and the production environment, the adoption of formal and tool-supported check-in/ check-out procedures for items, and management of a repository of assets/ utilities/ programs for their re-use across services.

An Object-Oriented approach to CM

The ONION Configuration Management strategy was based on an Object Oriented approach. This relies on the consideration that every entity implemented during the development of Web projects must be treated as an object. Moreover, every action that can be performed on that object must be considered as a method applicable to the object.

This strategy very well fits in the complex world of entities that the Web based projects must be able to manage. Such strategy allows also a high degree of flexibility because it is not mandatory to highlight in the very beginning the whole set of objects which will be used; rather it is possible to start with a small set of objects and methods and to take advantage of the possibility to define new objects and methods every time that this is needed.
Every entity belongs to a basic class called "Asset". This class owns a set of properties that are inherited by every object descending from the class, namely: Owner, Description, Location, Class, Version number, Construction date, Verification date, Approval date, Responsible person, Copyright, Access Control List, Configuration Control method. On top of this basic class, others were defined, including: Order; Web Site; Program; Document; etc.

Documentation management

Concerning document control, an integrated environment was adopted [LIBR], fully aligned with ISO 9000 requirement. This modular system was very useful in the light of setting-up a QMS since it provides extensions for the management of several ISO 9000 requirements (e.g: tool calibration, keeping of Quality Records, Supplier List management, etc.). In particular the tool-kit was very easy to introduce thanks to its strong integration with the Microsoft Office environment for the drafting of documents.
The most relevant features that proved to be suited for the specific environment were:

- configuration identification (the definition of codes for the unambiguous identification of document and their related version allowed to constantly know the state of each document);
- document management (handling several classes of documents, including: internal documents, external documents, fax, letters, contracts, Quality Records, etc.);
- a review and approval scheme based on a four-steps mechanism;
- a document matrix (identifying the responsibilities for preparation, review and approval of documents);
- the management of document distribution (for sake of confidentiality and security);
- the automatic management of document status, version, change history and authors;
- the management of company templates for all standard operating procedures and forms to be used in the company;
- meeting management (with tracking to closure of all action items);
- tool calibration (keeping under control all the calibration activities);
- time management (computing effort spent by project, work centres and work category);
- supplier management (keeping a supplier accredited list, with all relevant information);
- non conformity management (opening and tracking to closure of corrective actions).

An Intranet based Quality Management System

The basic architecture

The idea of setting-up a company Intranet designed for the management of the Quality Management System came across by observing that new technologies lead to more efficient and dynamic communication and thus to new infrastructures and work procedures for projects. Moreover, business process models are designed, information systems are established on Hypermedia platforms, and total quality management (TQM) gets a new vision when all effective processes of the organisation are made visible and accessible to the employees via Hypermedia and Internet systems.
Henceforth it was decided to manage the Quality Management System with a WWW support, as part of the company-wide Intranet.

This involved the porting of documents in the WWW environment, the definition of access rules, the creation of hyper-textual links across objects, the creation of modules through electronic forms, the linking with the mail systems etc.

Intranet as the future for QMS environments

The main advantages of a WWW-based Quality Management Systems have been experienced to be the following:

- availability at large: all people can have a direct, user-friendly access to all the items of the QMS;
- traceability: the hyper-textual mechanisms embedded in the WWW are particularly suited for managing references within and across documents of the QMS, allowing to browse through the complex structure of a Quality Management System and to keep under control the overall architecture of the system;
- maintainability: only the most recent version of QMS documents is always available on-line;
- distribution: thanks to access control list, the distribution of controlled copies is greatly facilitated;
- deployment: the availability of on-line forms (e.g: for tool procurement, anomaly management, training registration, supplier evaluation, etc.) is a powerful support to the deployment of the defined practices in the daily routine work;
- effectiveness: the integration with the development environment (e.g.: templates linked with the appropriate word processor, forms linked with the appropriate e-mail for posting) provides a straightforward way to information circulation within the company.

The experience has proven to be very positive and thus now the company considers the WWW as the principal environment for the development, tuning and maintenance of the QMS, from where to automatically derive the few paper copies that still are needed for the certification/ surveillance audits.

The Impacts and the Experiences gained

Quantitative goals and measurement plans

The following activities were foreseen for tracking the effectiveness of the Software Process Improvement Program:

- process assessment;
- process metrics;
- analysis of Return On Investment.

As far as quantitative measures are concerned, a small core set of basic metrics directly related to the business goals of the company was collected, including: projects that did not benefit of the improvement actions, projects in which additional practices were piloted and projects in which the new practices had become daily routine work.

In so doing, it was possible to have a considerable data set upon which a management-by-metrics activity was performed. The set of indicators collected is summarised in the following, together with the quantitative goals defined (values have been set basing on the experiences gained from earlier projects).

Indicator	Definition	Target Goal
Timeliness for the customer	Planned service development time/ actual time	> 80%
Re-use	% of common software modules re-used	> 50%

Fault density	Faults/ KLOC	< 1
Testing effectiveness	Faults in testing/ Total faults	>80%
Software Productivity	LOC/ person-month	>250
Asset Productivity	Html pages/ person-months	>150

Tab. 2 - Quantitative indicators

Impacts at company level

Results and lesson learnt are considered in the following from the following perspectives:

- technical;
- business;
- organisational;
- cultural.

Technical impacts

From a technical point of view, the following main achievements can be stated:

- selection and procurement of tools;
- definition of a draft Quality Manual;
- definition of guidelines for testing and CM;
- deployment of enhanced practices from the pilot projects to the daily routine work;
- performance of training and internal dissemination activities.

From a software engineering point of view, the following can be stated:

- the introduction of more systematic testing methods and tools is of paramount importance for level 1 SMEs and can be done with success in short time;
- the introduction of configuration management requires more care, both from a methodological and a cultural point of view;
- the development of WWW based multimedia applications cannot be ruled under a classical waterfall model but rather requires fast prototyping and Rapid Application Development approaches, that the company is setting as the next target for improvement.

Business impacts

From a business point of view it was perceived that the adoption of more mature software development/ maintenance practices results in increased confidence by the customers, which is expected to be reflected in positive returns from the market.
Moreover, the following lesson has been learnt: whereas customers are willing to reckon a direct value for testing activities, this is not always the case for configuration management activities which are normally considered as an unavoidable overhead which should not be charged onto them; this pushes very much on the adoption of approaches whose cost/ benefit ratio is positive from a productivity and timeliness point of view.

Organisational impacts

The project was run in co-operation between the communications/ technologies department and the consulting department, which has specific skills and experiences in software process improvement. Due to this organisational peculiarity, the project allowed to transfer internally the process improvement culture previously owned only by a subset of people. This resulted in higher company integration, which is a key element for the strategic projects that the company

runs with major customers, requiring a strict combination of technical and consulting capabilities.
Moreover, due to the several interactions across people, the project facilitated a more clear definition of roles within the company. At the same time, the company is more and more experiencing the adoption of a paradigm oriented towards a "flexible resource pool" with dynamic allocations and resource sharing across projects, rather than the permanent assignment of technical staff to a predetermined area/ group. This, albeit being more complex from a resource management point of view (which is more and more done at company level, letting the detailed activity planning/ tracking at Project Level), is felt as very fit for SMEs confronting with a fast changing market. To this respect, PI^3 has contributed to the strengthening of the positive mood and feel that is needed in a company willing to adopt a "resource pool" approach.

Cultural impacts

Involvement of the people was positive, without major resistance in adopting new tooling and methods; a clear evidence of this is the fact that Project Leaders, besides their involvement in the pilot projects, autonomously decided to apply the new testing methods also to products already in field, for sake of sanity/ regression checking.
Still some barriers are present in the application of more rigorous configuration management methods and in test execution tracing/ problem report management. This is due to the fact that such activities are sometimes foreseen as a project overhead which does not bring to tangible results in short terms and there is the risk that they get assigned a low priority when time schedule pressure is high.
In order to overcome this problem, we are looking at the definition of WWW based support forms in order to facilitate the uptake of such practices by adopting a work style that is already familiar and accepted by the technical staff.
Another way to attack this issue is to stress internal dissemination of the enhanced practices by means of workshops held by the people who experienced the new solutions (this is felt much more convincing for the developers than a "theoretical" tutorial).
Moreover, an effort is undergoing to consider such activities in the light of continuous improvement, with a medium-long term plan consisting in ISO 9001 certification, which represents for the company a challenging goal.
Several additional skills have been acquired by the project staff as a result of the experiment (e.g.: high level knowledge of ISO 9000 and Software Process Improvement principles; knowledge of testing methods and techniques; knowledge of configuration management principles; in-depth knowledge of the procured tools); the nature of the experiment implied also considerable changes for the professionals involved, in terms of their way of working, their skills and disciplines, etc.
The impact of the experiment on human factors can be summarised as follows:

- people showed enthusiasm in using new tools;
- people accepted the idea of systematic testing (not just debugging) and independent verification and validation;
- people positively experienced the usefulness of project guidelines, provided these are pragmatic and built as much as possible bottom-up from the hands-on experience;
- people were a bit reluctant on the adoption of more rigorous activity tracing methods, when this was not felt as directly contributing to the project technical needs: "bureaucracy" is not welcome.

Experiences achieved and future goals

This section provides a summary view on the usefulness of the Process Improvement program itself, identifying the strengths and weaknesses of the adopted approach and the overall benefit for the organisation.
Strong aspects can be considered the following:

- deployment in two pilot projects;
- involvement of people from different departments;
- combination of technical and methodological aspects.

The following aspects show room for improvement:

- parallel activation of two improvements (testing and CM) onto a small organisation at a time; this is good at rule definition level but is not easy at project level, where improvements have to be managed with care in order not to overwhelm the project staff that has to keep in any case the planned goals and schedules;
- management of project guidelines within an overall framework; this was not foreseen at the beginning but resulted soon to be a need for the company that thus defined a first draft Quality Manual adherent to ISO 9001 before getting to the definition of detailed guidelines.

In the overall, the PI^3 project is felt as very successful; nevertheless, if we were to repeat it, we would make specific changes to our approach in order to overcome the two identified weaknesses, mainly: more accurate timing of deployment of improvements in the daily routine work and definition of company rules adopting a top-down approach.
The issue of Process Improvement deployment is felt as critical, since for SMEs intensive software process actions might bring to disruption of the normal company activities. It is our opinion that this should not happen, provided that the improvement actions are seen as a significant part of a global Process Improvement approach that the company should continuously apply.
In any case, in order to avoid risks, the following provisions can be put forward:

- care has to be taken in the detailed planning of the improvements in order to avoid the overlapping of the most effort consuming activities;
- improvements shall be extended to other projects adopting a bottom-up approach, that is to say introducing additional practices in a controlled way and under the responsibility of the Project Leader, only after they have been discusses with and accepted by the involved designers;
- the measurement of quantitative results obtained by the pilot projects has to be the major criteria for deciding the deployment of additional practices into the normal company activities;
- internal training and dissemination must have a big emphasis.

Acknowledgements

The PI^3 Project has been run under the auspices of the CEC DG III within the scope of the ESSI (European Software and System Initiative) Program [ROHE] of the ESPRIT Fourth Framework. This support has proven to be extremely important in ensuring the overall success of the initiative, which lies in the mainstream of the company core business.
In particular we have to thank the European Commission Officer, Mrs. Annalisa Bogliolo, for the continuous and competent support offered in the set-up and running of the Project.

References

[BACH] R. Bache, G. Bazzana
"Software Metrics for Product Assessment"
McGraw Hill, London, 1995, International Software Quality Assurance Series, ISBN 0-07-707923-X

[BAZZ] G. Bazzana, E. Fagnoni, M. Piotti, G. Rumi, F. Visentin
"Testing in the Internet"
EuroStar 96, 4th European Conference on Software Testing, Analysis and Review, Amsterdam, December 1996

[BEIZ] B. Beizer
"Software Testing Techniques"
Van Nostrand Rheinhold, 1990

[BOWE] N. Bowers
"WebLint: Quality Assurance for the World Wide Web"
Proceedings of 5th International WWW Conference, Paris, May 1996
pagg. 1283-1290

[CEC1] Commission of the European Communities
"Information Technology Security Evaluation Criteria (ITSEC)"
Version 1.2, CEC, 28 June 1991

[CEC2] Commission of the European Communities
"Information Technology Security Evaluation Manual (ITSEM)"
Version 1.0, CEC, 10 September 1993

[IMAG] ImagiWare
"Doctor HTML"
http://imagiware.com/RxHTML.cgi

[LIBR] Libra
"Gest Doc"
http://www.smartcom.it/libra/gestdoc.html

[MARK] D. M. Marks
"Configuration Management"
McGraw-Hill, 1996

[ROHE] M. Rohen
"ESSI: Objectives, Strategy and Implementation of Software Best Practice Actions"
Proceedings of ISCN 95 Conference, Wien, September 1995

Description and Evaluation of the SPICE Phase One Trials Assessments*

Khaled El Emam[a]
Fraunhofer - Institute for Experimental Software Engineering
Sauerwiesen 6, D-67661 Kaiserslautern
Germany
elemam@iese.fhg.de

Dennis R. Goldenson[b]
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
USA
dg@sei.cmu.edu

Abstract

The SPICE (Software Process Improvement and Capability dEtermination) project aims to deliver an international standard on software process assessment. Unique among software engineering standards efforts, there is an empirical trials activity as part of the SPICE project. The first phase of these trials was completed in calendar year 1995. During phase one, data was collected to evaluate design decisions of the SPICE framework and the usability of the core SPICE documents. In this paper we describe these assessments and present an evaluation of some of the documents based on 35 assessments conducted during phase one of the SPICE trials. The results indicate that the SPICE framework is in general sound, but they also highlight some potential weaknesses. Since the completion of these trials, the SPICE documents have been revised taking the trials results into account.

Introduction

The objective of the SPICE (Software Process Improvement and Capability dEtermination) Project is to deliver an ISO standard for software process assessment. More information about SPICE may be found in [1][2][3][7][9][10]. As part of this project there are a set of empirical trials [4][8]. These empirical trials are scheduled to be completed in three broad phases. The first phase was completed in calendar year 1995. Its results were based on several sources of data, including a series of questionnaires completed by both assessors and assessees from 35 assessments conducted world-wide, project problem reports and change requests, and the actual rating profiles forthcoming from the assessment. The focus of phase 1 was on evaluating the design decisions of the SPICE framework, and the usability of version 1.0 of the core SPICE document set[6]. The results from phase 1 were used to help identify strengths and shortcomings, and inform decisions about the content of the document set prior to standardization.

In this paper we describe the assessments conducted during phase 1 of the SPICE trials and present an evaluation of the core document set from the experiences of 35 assessments. This is based on the work done by the authors and reported in the phase 1 trials final report [11]. The results indicate that the SPICE model and rating framework are in general sound and have been found to be useful and usable, but they also highlight some potential weaknesses. As of this writing, the weaknesses have been taken into consideration in developing version 2 of the SPICE document set.

In the next section we describe the method that was used for data collection and data analysis. Section 3 presents the detailed results. Section 4 concludes the paper with a summary and with an overview of the subsequent phases of the SPICE trials.

---

* The views stated in this paper are those of the authors and do not necessarily reflect the positions of their employers or their funding agencies.

[a] The work reported in this paper done by El Emam in the SPICE project has been supported, in part, by the Applied Software Engineering Centre (ASEC) in Montreal.

[b] The Software Engineering Institute is sponsored by the U.S. Department of Defense.

[6] Some assessments in phase one of the trials also used earlier versions of the documents than version 1.0.

Research Method

The SPICE trials are a collaborative effort amongst a substantial number of people around the world. In order to manage the data collection effort on a global scale, four regional centers were set up to coordinate data collection in each region. These regions were: the Pacific Rim, Europe, Canada, and the USA In most regions, assessments using the SPICE documents were conducted in 1995. During each of these assessments a set of questionnaires[7] were administered. For the purposes of this paper, we obtained responses from two groups of people: (i) lead assessors who were in charge of the trials assessments, and (ii) the sponsors of the assessments in the Organizational Unit (the organization or part of the organization that was being assessed). These give us the assessors' and assessees' perspectives respectively. In total, questionnaire data from 35 assessments were collected before the response deadline. Of these, 20 were conducted in Europe, 1 in Canada, and 14 in the Pacific Rim.

Document Name[8]	Brief Description
Baseline Practices Guide (BPG)	This document defines at a high level the fundamental software activities and their capabilities structured in 5 levels. The activities are divided into five process categories: customer-supplier, engineering, project, support, and organization.
Assessment Instrument Guide (AIG)	Defines the requirements for a conformant assessment instrument.

Figure 1: Brief description of the core SPICE documents whose evaluations are presented in this paper.

The objectives of our analysis of the questionnaire responses as presented in this paper are twofold. First, to describe what actually happened during the phase one assessments. Second, to present an evaluation of some of the core SPICE document set. The evaluation results for the documents described in **Figure** are presented in this paper.

The results presented here are the percentage of responses to various questions[9]. These results are shown in the form of histograms. To evaluate the documents, we identify the proportions of respondents who are supportive (as opposed to critical) of either the SPICE design decisions or the claim that the documents are usable. A supportive response is one:

- that says something positive about SPICE, and/or
- that will *not* require any changes to the draft SPICE documents (i.e., the ones that were used during the phase one trials assessments)

We have also made a distinction between „very supportive" and „moderately supportive" responses. This distinction helps make clear the extent of support for SPICE. For example, assume that a question asked the respondents to express their extent of agreement to the statement „The assessment improved awareness of software process issues among the organizational unit's software engineers", and that it had the following four response categories: „Strongly Agree", „Agree", „Disagree", and „Strongly Disagree". As shown in **Figure** , the „Strongly Agree" and „Agree" responses would be considered supportive of SPICE, and the „Disagree" and „Strongly Disagree" responses would be considered to be critical of SPICE. Furthermore, the „Strongly Agree" response category would be considered to be „very supportive" of SPICE and the „Agree" response category would be considered to be „moderately supportive" of SPICE.

Supportive Responses	Critical Responses

[7] Copies of these questionnaire may be obtained directly from the authors.

[8] The names, structuring, and organization of the SPICE documents have changed since the completion of phase 1 of the SPICE trials. In this paper, however, we will continue to refer to the documents as they were used during phase 1 of the trials.

[9] An inferential analysis of this data has been performed and is presented in [5].

Very Supportive Responses	Moderately Supportive Responses	
Strongly Agree	*Agree*	*Disagree* *Strongly Disagree*

Figure 2: Types and examples of response categories.

For the histograms presented in the results section of this paper, it will be noticed that the number of responses differ considerably. The reasons for this are mainly that: (i) for different questionnaires, we received a different number of responses, (ii) for different questions, there were different numbers of missing data (i.e., the respondent did not answer the question at all), and/or (iii) responses to particular questions excluded the respondent from the analysis for other questions (e.g., if the assessor did not use the Assessment Instrument Guide for preparing and/or during an assessment, then that assessor was excluded in the analysis of certain questions that assume that the Assessment Instrument Guide was used).

It must also be recalled that such an extensive empirical evaluation of a software process assessment framework and/or model, as being conducted in the SPICE trials, has not been conducted before. Therefore, it is difficult to compare the trials' results to those from previous studies. In the few exceptional cases where precedents exist, this comparison has been made in the presentation of our results.

Results

Description of the Assessments

The SPICE documents provide general guidance for conducting assessments. The activities defined in the documents[10] are summarized in **Figure**. As seen in **Figure** , most of the assessment activities defined in the SPICE documents were performed during most of the assessments. However, in less than 70% of the assessments were the „verify existence ratings" and „determine derived ratings" performed. The former may be due to the fact that existence ratings were performed in a smaller number of assessments (when compared with adequacy ratings).

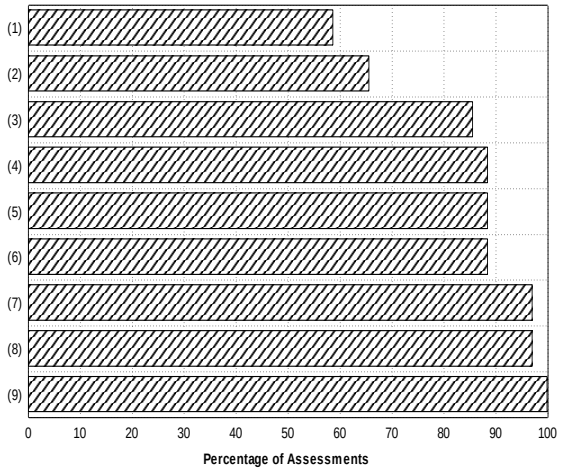
During the assessments, the most commonly used type of assessment instrument was a paper based checklist followed by a computerized spreadsheet (**Figure**). Apart from the spreadsheet, it was rare that any other form of computerized instrument was used. The instruments that were used were developed mostly by the lead assessors themselves (**Figure**). Very few of the assessors (only 35 percent) used the exemplar instrument provided by the SPICE project.

Most of the information that was collected during the assessments was through interviews followed by the review of documents or interim work products (**Figure**). No assessors used assessee self-reports, and very few collected data prior to the on-site visit (12%).

[10] We have excluded „Collecting and verifying information" because it is a basic activity that has to be performed in any assessment.

Activity	Brief Description[11]
Define and review the assessment inputs	The assessors should review the assessment purpose, scope and constraints to ensure that they are consistent and that the assessment purpose can be fulfilled, responsibilities, and any extended process definitions.
Select process instances	This involves mapping the organizational unit's processes to the BPG model and then selecting instances in a manner that would satisfy the assessment purpose.
Identify assessment risk factors	Risk factors could include changes in commitment of the sponsor, unplanned changes to the structure of the assessment team, organizational changes, and lack of confidentiality.
Brief the Organizational Unit's personnel	The briefing should include an overview of the assessment purpose, scope, and constraints, the conduct of the assessment, and how the assessment outputs can be used to provide the most benefit to the organization.
Verify ratings	Supporting documentation and records are collected to verify the ratings made.
Determine Derived Ratings	Derived ratings are based on an aggregation of actual ratings for process instances.
Validate the ratings	This would include comparing the results with those from previous assessments of the same organizational unit, looking for inconsistencies in the ratings of related processes, and feedback sessions of preliminary findings to the organizational unit.
Present the results to OU management	The assessment findings are presented to organizational unit management and the sponsor.

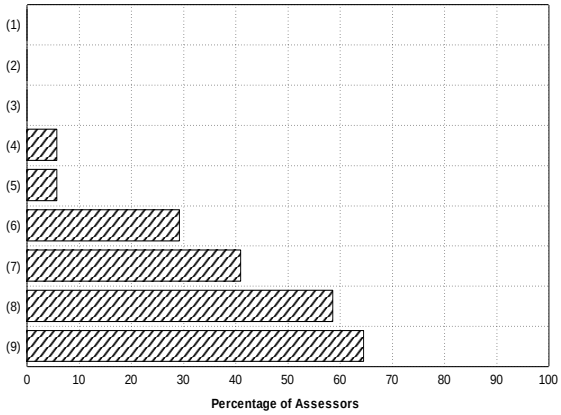
Figure 3: Brief description of some assessment activities.



No.	Activity	Percentage of Assessments
(1)	Verify existence ratings (existence is a two-point rating scale that can be used in an assessment)	(20/34) = 59%
(2)	Determine derived ratings	(23/35) = 66%
(3)	Identify assessment risk factors	(30/35) = 86%
(4)	Verify adequacy ratings (adequacy is a four-point rating scale that can be used in an assessment)	(31/35) = 89%
(5)	Present the assessment results to OU management	(31/35) = 89%
(6)	Validate the ratings	(31/35) = 89%
(7)	Select process instances	(34/35) = 97%
(8)	Brief OU personnel	(34/35) = 97%
(9)	Define and review the process inputs (i.e., purpose, scope, constraints, responsibilities, and extended process definitions)	(35/35) = 100%

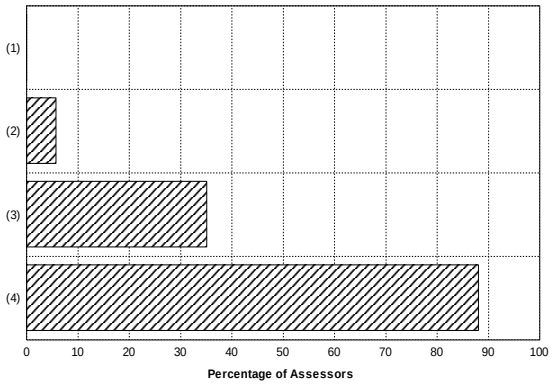
Figure 4: The activities that were performed during the assessments.

[11] These are only brief descriptions of the activities to aid the reader in interpreting the charts. More details of the activities are in the SPICE documents.



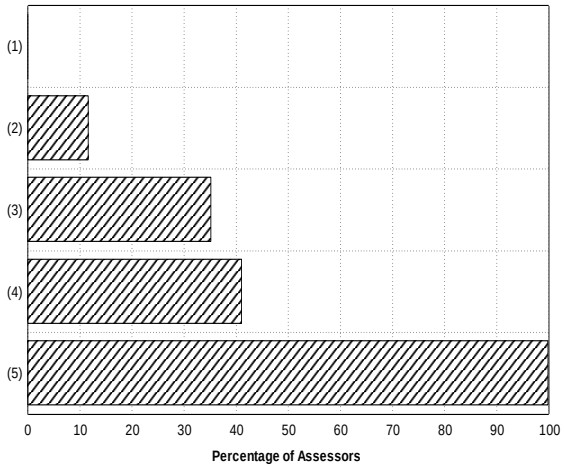
No.	Type of Assessment Instrument	Percentage of Assessors
(1)	Computer Based Flat File	(0/17) = 0%
(2)	Computerized Checklist	(0/17) = 0%
(3)	Computerized Expert System	(0/17) = 0%
(4)	Computerized Questionnaire	(1/17) = 6%
(5)	Computerized Relational Database	(1/17) = 6%
(6)	Computerized Scoring	(5/17) = 29%
(7)	Paper Based Questionnaire	(7/17) = 41%
(8)	Computerized Spreadsheet	(10/17) = 59%
(9)	Paper Based Checklist	(11/17) = 65%

Figure 5: Type of assessment instruments used by the assessors.



No.	Developer(s) of the Assessment Instruments Used	Percentage of Assessors
(1)	Organizational Unit representative(s)	(0/17) = 0%
(2)	Third party tool builders / vendors	(1/17) = 1%
(3)	Supplied as an exemplar from the SPICE project	(6/17) = 35%
(4)	Experienced assessors(s)	(15/17) = 88%

Figure 6: The developer(s) of the assessment instruments that were used.



No.	Method for Collecting Information	Percentage of Assessors
(1)	Assessee self reports	(0/17) = 0%
(2)	Data collection prior to on-site visits	(2/17) = 12%
(3)	Group Feedback sessions	(6/17) = 35%
(4)	Document or interim work product reviews	(7/17) = 41%
(5)	Interviews	(17/17) = 100%

Figure 7: The method(s) that the assessors used to collect information during the assessment.

Overall Evaluation

The assessment sponsors' overall perceptions are generally quite positive towards SPICE. Almost *all* of them agreed that the benefits of their assessments were at least "on balance" worth the expense and time their organizations expended (**Figure**); almost 40 percent said their assessments were "more than worth the expense." Almost 80 percent of the assessment sponsors agree that awareness, "buy-in," and support for process improvement improved among their organizations' management as a result of their assessments. However, only 65 percent agree to a similar question about their technical staffs, and relatively few chose the "strongly agree" response option to either question about commitment to SPI resulting from the assessments.

Overall, the experienced assessors are somewhat more positive towards SPICE than are the assessees, but they too tend to qualify their responses (**Figure**), indeed more so than do the assessees. Almost all of the assessors say that the organizational unit personnel were satisfied with the results of their assessments; over 80 percent think that the assessments improved awareness of SPI issues among the engineers in the organizational units that were assessed. Perhaps most pertinent from a SPICE perspective, 85 percent of the experienced assessors characterize the SPICE approach as being at least somewhat better than "other assessment methods[12]" with which they are familiar. Once again, though, relatively few of their answers (less than 25 percent to all three questions) fall into what we have classified as responses that are "very supportive" of the SPICE document set and the phase 1 trial assessments.

Accuracy of Assessment Results

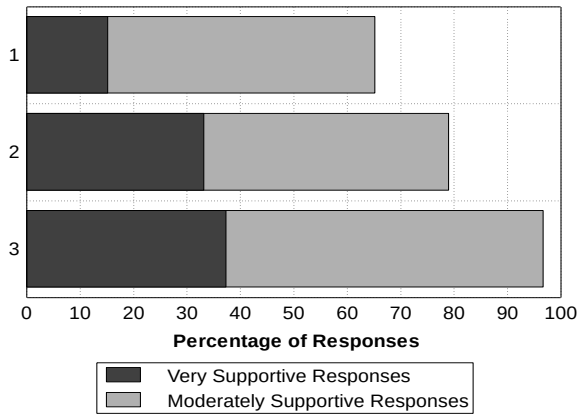
As seen in **Figure** , the assessment sponsors are generally quite satisfied with the accuracy and actionability of their assessment results. Over 90 percent of the assessors report that their assessments provided valuable direction for process improvement in their organizations, characterized their organizations' strong points at least "reasonably well," and that their SPICE process profiles accurately described their organizations' major problems. Once again, though, the assessees do express some reservations. Over 20 percent of them say that the process profiles were only "generally accurate" within the scope of their assessments. Well over 30 percent of the assessment sponsors report inappropriately identified "problems" in their process profiles; a similar proportion say that their profiles *failed* to identify problems in the scope of their assessments.[13]

[12] SPICE does not define a complete process for conducting an assessment. It does provide guidance however.

[13] There may be question wording problems though. „Any“ problems and „anything“ are quite unrestrictive modifiers.

We can compare some of the results obtained for SPICE with those obtained in another survey[14] of users of the CMM [6]. When asked about how well the CMM assessment described the organization's major problems with the software process, 98% responded with the „very accurately" or „generally accurately" categories. This is comparable to the 91% obtained from the assesees in the SPICE trials. In addition, when asked how well the assessment characterized the organization's strong points, 92% of the respondents to the CMM survey chose the „very well" or „reasonably well" response categories. This percentage is comparable to the 93% obtained from the SPICE questionnaires. Therefore, at least by these two criteria, the results from the phase 1 assessments are comparable to those obtained from a previous process assessment model survey.

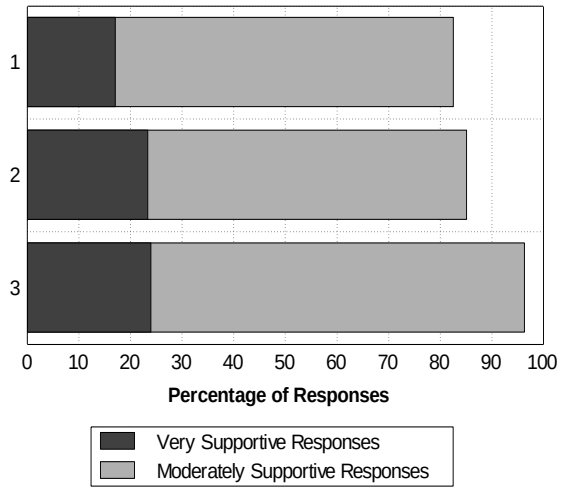
Here, the assessment sponsors are more supportive of SPICE in its phase 1 incarnation than are the assessors. **Figure** summarizes the assessors' responses to two general questions about the accuracy and actionability of the phase 1 assessments. The "supportive" responses do approach 80 percent in both instances. However, rather few of the assessors chose the unequivocal response option ("strongly agree").



No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	The assessment improved awareness, „buy-in," and support for process improvement among the organization's technical staff	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(17/26) = 65%
(2)	The assessment improved awareness, „buy-in," and support for process improvement among the organization's management	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(19/24) = 79%
(3)	Were the benefits of the assessment worth the expense and the time expended?	• More Than Worth the Expense • On Balance Worth the Expense	• Not Worth the Expense	(31/32) = 97%

Figure 8: Overall evaluation of the SPICE assessment by the assessees.

[14] This survey was done at least one year after the assessment was conducted.



No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	The assessment improved awareness of software process issues among the organizational unit's software engineers	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(24/29) = 83%
(2)	Compared to other assessment methods with which you are familiar, how would you characterize the SPICE approach?	• SPICE Is Much Better • SPICE Is Better on Balance	• SPICE Is Much Worse • SPICE Is Worse on Balance • SPICE Is Neither Better Nor Worse	(29/34) = 85%
(3)	The organizational unit's personnel were satisfied with the results of the assessment	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(28/29) = 97%

Figure 9: Overall evaluation of the SPICE assessment by the assessors.

General Evaluation of the BPG

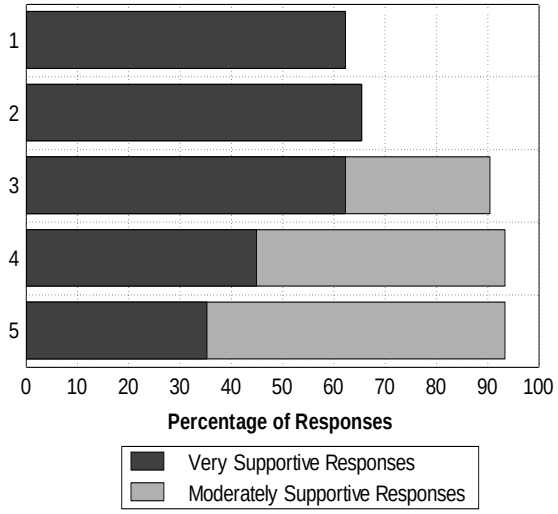
The BPG document describes the SPICE architecture. The SPICE architecture is two dimensional. Each dimension represents a different perspective on software process management. One dimension consists of *base practices*. A base practice is defined as a software engineering or management activity that addresses the purpose of a particular process. Base practices are grouped into *Processes*, which in turn are grouped into *Process Categories*. An example of a process is *Develop System Requirements and Design*. Base practices that belong to this process include: *Specify System Requirements*, *Describe System Architecture*, and *Determine Release Strategy*. The other dimension consists of *generic practices*. A generic practice is an implementation or institutionalization practice that enhances the capability to perform a process. Generic practices are grouped into *Common Features*, which in turn are grouped into *Capability Levels*. An example of a Common Feature is *Disciplined Performance*. A generic practice that belongs to this Common Feature stipulates that data on performance of the process must be recorded. Base and generic practices can be rated during an assessment.

The experienced assessors express reservations about this version of the BPG after having used it in their phase 1 assessments (**Figure**). *All* of them do agree that the BPG was in fact useful for the assessments about which we queried. However fewer than half of them chose the more strongly worded response alternative. Two-thirds of the assessors said that additional processes or common features should be included in the BPG. Very few agree that the BPG provides sufficient direction for scoring the practices.[15]

The assessees tend to have generally positive attitudes towards the BPG (**Figure**), although they would not have used it as extensively as the assessors. Almost seventy percent stated that there were no important missing areas in the BPG (question 1). The BPG does allow for extending the practices through the generation of application/sector specific practice guides. The majority of assessees felt that the process improvement order implied in the SPICE framework was valuable. Almost all of the

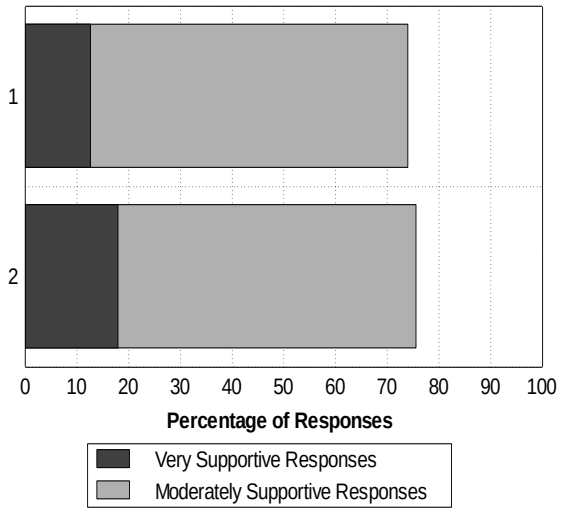
[15] This question may overstate dissatisfaction with the BPG. It could be interpreted to mean that the BPG does not adequately define the practices. It could also mean that scoring is the domain of another SPICE document: the Process Assessment Guide, which explains how to rate practices. Therefore, this result does not necessarily indicate that the BPG was not achieving its purpose.

assessees (ninety two percent) felt that the BPG provides real hope for long term process improvement.



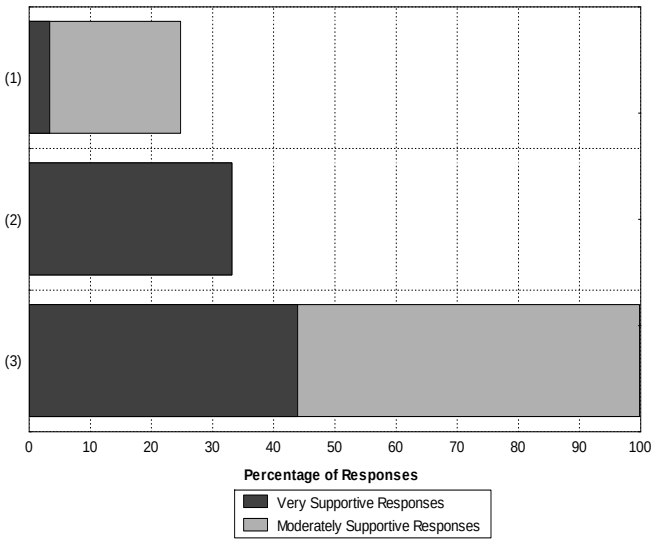
No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	Did the software process profile inappropriately identify anything as a problem(s)?	• No	• Yes	(20/32) = 62%
(2)	Did the software process profile fail to identify any problems within the scope of the assessment?	• No	• Yes	(21/32) = 66%
(3)	To the best of your knowledge, how accurately did the software process profile describe the organization's major problems within the scope of the assessment?	• Very Accurately • Generally Accurately	• Not Very Accurately	(29/32) = 91%
(4)	How well did the assessment characterize the organization's strong points?	• Very Well • Reasonably Well	• Not Very Well	(29/31) = 93%
(5)	The assessment provided valuable direction about the priorities for process improvement in the organization	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(29/31) = 93%

Figure 10: Assessees' impressions about the accuracy of the assessment results.



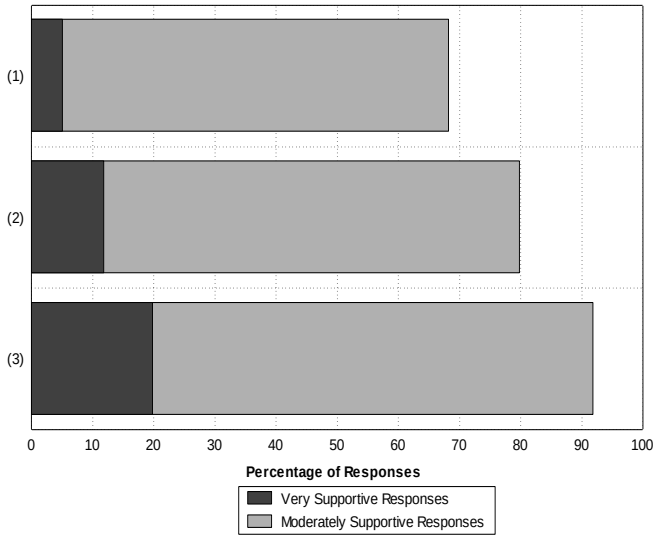
No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	The assessment provided valuable direction about priorities for process improvement in the organizational unit	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(23/31) = 74%
(2)	The assessment helped management identify important strengths and weaknesses in their organizational unit	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(25/33) = 76%

Figure 11: Assessors' impressions about the accuracy of the assessment results.



No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	The BPG provides sufficient direction for scoring the practices	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(7/28) = 25%
(2)	In your opinion, are there any processes or common features which are not covered in the Baseline Practices Guide and should be?	• No	• Yes	(8/24) = 33%
(3)	Overall, the BPG was useful for this assessment	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(34/34) = 100%

Figure 12: Assessors' overall evaluation of the BPG.



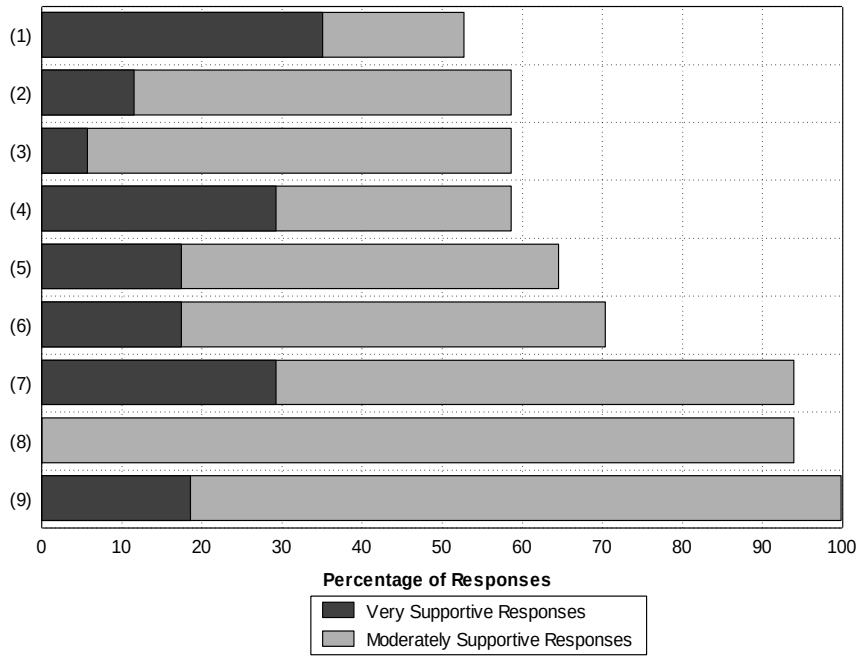
No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	There are important areas that the BPG does not address	• Strongly Disagree • Disagree	• Strongly Agree • Agree	(13/19) = 68%
(2)	The SPICE Baseline Practices Guide provides valuable direction about the order in which process improvements should be made	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(20/25) = 80%
(3)	Because of its comprehensive nature, the BPG provides real hope for long term process improvement	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(23/25) = 92%

Figure 13: Assessees' overall evaluation of the BPG.

Evaluation of the AI Guide

Much of the analyses presented thus far were based on questionnaires meant to be collected at the end of each of the phase 1 assessments. The AI Guide was not available throughout most of phase 1, and there was concern about over-taxing the good will of the assessors. Hence the following analyses are based on a single questionnaire that was created for distribution to each experienced assessor after the completion of all of his or her phase 1 assessments.

The experienced assessors' responses to nine general questions about the AI Guide are summarized in **Figure** . Their reviews are somewhat mixed. First of all, notice that large majorities agree that the AI Guide meets its most basic requirements (questions 7, 8, and 9). They agree that the Guide does in fact provide useful help for developing an assessment instrument, that the coverage of the indicator set is adequate, and that the Guide is compatible with the other two core SPICE documents. However, fewer (71 percent) think that the guide is helpful for selecting an existing assessment instrument (question 6), and fewer than two-thirds agree to a series of assertions (in questions 1 through 5) about the clarity and usability of the AI Guide.



No.	Question	Supportive Response Categories	Critical Response Categories	Percentage Supportive
(1)	Without the availability of an AI Guide an assessment is/would be more difficult to conduct	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(9/17) = 53%
(2)	The AI Guide is/would be usable in terms of time scale and effort for developing a new assessment instrument	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(10/17) = 59%
(3)	The AI Guide is/would be usable in terms of time scale and effort for selecting an assessment instrument	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(10/17) = 59%
(4)	Without the availability of an AI Guide an assessment is/would be more difficult to prepare for	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(10/17) = 59%
(5)	The AI Guide is clear and easy to understand	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(11/17) = 65%
(6)	The AI Guide is/would be helpful in selecting an Assessment Instrument	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(12/17) = 71%
(7)	The AI/Guide is/would be helpful in developing and assessment instrument	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(16/17) = 94%
(8)	The coverage of the indicator set in the AI Guide is adequate	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(16/17) = 94%
(9)	The AI Guide is compatible with the BPG and the Process Assessment Guide	• Strongly Agree • Agree	• Strongly Disagree • Disagree	(16/16) = 100%

Figure 14: Evaluations of the AI Guide overall by the assessors.

Conclusions

The SPICE trials do show that it is possible to provide empirical evidence that can inform decision making for an evolving, prospective international standard. In the spirit of continuous improvement, the phase 1 trials identified a number of areas in need of modification in the first version of the SPICE documents. As planned, the phase 1 of the SPICE trials was completed in time for a critical decision point in the standardization process of the SPICE document suite. This was a ballot by the member national bodies on the documents. The results from phase 1 of the SPICE trials were used as input into this process, whereby the phase 1 trials report was made available to all member bodies prior to

the ballot deadline. We are aware of at least two bodies who made explicit reference to the results of the trials in their comments.

Phase 2 of the SPICE trials is currently on-going. These trials will continue to evaluate the SPICE document set in actual industrial use. During this second phase, version 2.0 of the document set will be used. Version 2 of SPICE is also an ISO Preliminary Draft Technical Report (PDTR). The expectation is that the phase 2 trials will provide some further results in time for the PDTR ballot in 1997. Particular attention will be paid to:

- evaluating the criteria for establishing the conformance of process models and assessment methods to the SPICE framework (for this, different conformant assessment methods and models will be used);
- how well the SPICE processes and generic practices are grouped, and whether they are expressed in an appropriate order;
- evaluating the benefits of increased capability as measured by the SPICE Capability dimension;
- evaluating the extent to which different rating teams in fact make equivalent judgments; and
- evaluating the effort required for SPICE-based assessments

At this writing, we anticipate the participation of several hundred assessments in the phase 2 trials. Phase 3 of the trials will attempt to expand participation even wider, and will concentrate on gathering evidence about the business value of software process improvement using the SPICE framework.

Acknowledgments

Many individuals have contributed their time to the SPICE trials whom we cannot thank individually for lack of space. However, we would like to acknowledge the key contributions to the phase 1 trials of Fiona Maclennan, Mary Tobin, Gary Ostrolenk, Robin Hunter, Ian Woodman, Peter Krauth, Peter Marshall, Terry Rout, Greg Jenkins, Jerome Pesant, Dave Kitson, and Inigo Garro. We also wish to thank Dave Kitson and Jerome Pesant for their comments on an earlier version of this paper.

References

[1] A. Dorling: „SPICE: Software Process Improvement and Capability dEtermination“. In *Information and Software Technology*, 35(6/7):404-406, June/July 1993.

[2] J-N Drouin: „Software Quality - An International Concern“. In *Software Process, Quality & ISO 9000*, 3(8):1-4, August 1994.

[3] J-N Drouin: „The SPICE Project: An Overview“. In *Software Process Newsletter*, IEEE Computer Society, No. 2, pages 8-9, Winter 1995.

[4] K. El Emam and D. R. Goldenson: „SPICE: An Empiricist's Perspective“. In *Proceedings of the Second IEEE International Software Engineering Standards Symposium*, pages 84-97, August 1995.

[5] K. El Emam and D. R. Goldenson: „An Empirical Evaluation of the Prospective International SPICE Standard“. To appear in *Software Process Improvement and Practice*, 2(2), 1996.

[6] D. Goldenson and J. Herbsleb: *After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success*. Technical Report CMU/SEI-95-TR-009, Software Engineering Institute, 1995.

[7] M. Konrad: „On the Horizon: An International Standard for Software Process Improvement“. In *Software Process Improvement Forum*, pages 6-8, September/October 1994.

[8] F. Maclennan and G. Ostrolenk: „The SPICE Trials: Validating the Framework“,. In *Proceedings of the 2nd International SPICE Symposium*, Brisbane, June 1995.

[9] M. Paulk and M. Konrad: „Measuring Process Capability Versus Organizational Process Maturity“. In *Proceedings of the 4th International Conference on Software Quality*, 1994.

[10] T. Rout: „SPICE: A Framework for Software Process Assessment“. In *Software Process Improvement and Practice Journal*, Pilot Issue, pages 57-66, August 1995.

PRACTICAL TIPS ABOUT SPICE : Structure and Guide for Use

Jean-Martin SIMON
CISI - Région Rhône-Alpes
3, rue Maryse Bastié - BP 23 - 69675 LYON - Bron - Cedex - FRANCE
Phone : (33) 04.72.15.51.17 - Fax : (33) 04.72.15.51.15 - e_mail : simonjm@jupiter.cisi.fr

Abstract

After 3 years of intensive work and international collaboration, The future ISO/IEC standard for software process assessment has progressed toward a second version, baselined in May 96. The v2 of SPICE (Software Process Improvement and Capability dEtermination) gives to the software community a new model and a framework to assess the software processes. These components are now tested during the second phase of the SPICE Trials. To help those who want to use SPICE v2 as a tool for managing Quality, this paper describes the process model and the capability rating, and gives some suggestions and advice based on practical assessments feed-backs.

Key-words : *Software Process / Process Assessment / SPICE / Quality*

Introduction

In mid'93, the SPICE (Software Process Improvement and Capability dEtermination) project started to develop a working draft for the software process assessment standard on behalf of ISO. A model for assessing the software processes, as well as guidance documents for process assessment, process improvement and capability determination were established in June 1995 as the first version. They correspond now to the so-called *SPICE version 2* which has been delivered as Technical Reports to ISO during the end of 1996. Refer to [IE 97] for a full description of the SPICE project and deliverables.

At the same time, trial phases have been planned in order to experiment the outputs of project as soon as possible. This original approach in the standardization process consists in 3 Phases. The software community was asked to experiment the process model and the assessment framework during the Phase 1 Trials. The results are analyzed in [SP95], [WO96], [EL96], and [MA96]; they have been largely used to undertake the evolution of the process model for version 2.

The Phase 2 Trials has now started on the basis of the new version of SPICE. This gives again the opportunity to experiment the future standard and to contribute to the improvement of the current process model and assessment framework described in the SPICE document set.

In this paper, we first give an overview on the SPICE components. Then the results of SPICE assessments experiments are then described, in terms of feedback to give some practical recommendations and advises.

SPICE's Key Concepts

In this chapter, we only give some of the main characteristics about SPICE to define the context of the assessment experiment. The reference documents and better, the original SPICE document set give a full understanding of the SPICE assessment approach. As for other model for software process management, SPICE considers that the assessment process is one of the fundamental means to manage software quality improvement and supplier's capability determination. The Figure 1 shows that the processes are assessed by a *qualified* assessor according to a *model* using some *guidance*. The results of the assessment are used as inputs for :

- ◊ process improvement : an initial assessment gives the initial capability of the process and shows its strengths and weaknesses; later the results of the improvements initiative are confirmed with an other assessment.
- ◊ capability determination : the assessment results allows the customer to identify the risk and the capability of the supplier.

In Figure 1, *Part x* reefers to the SPICE document set (see below 2.1).

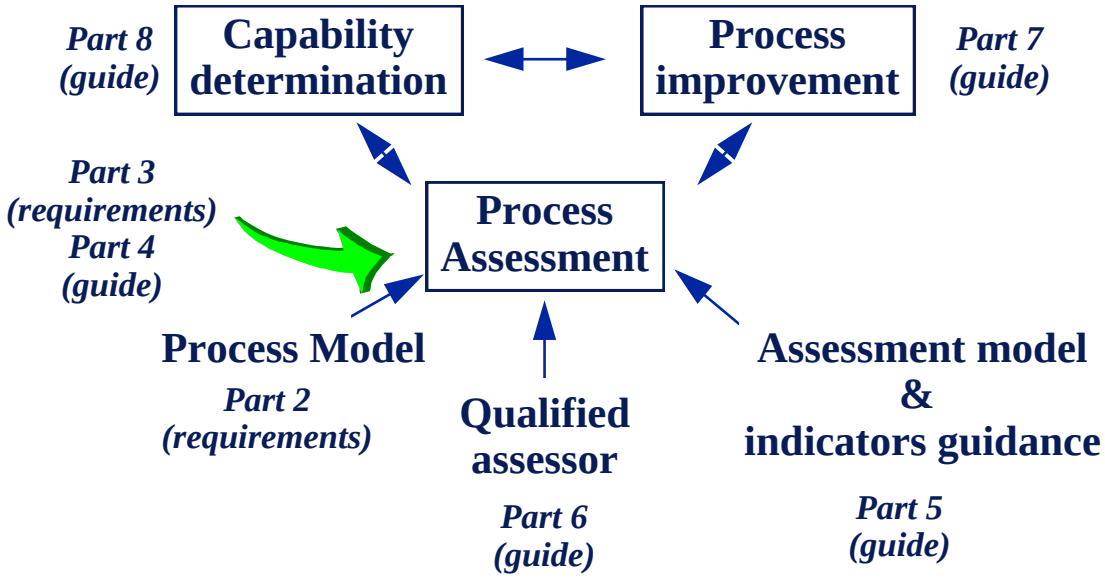


Figure 1 : SPICE framework and components

SPICE is not only a process model [IS96a]. For satisfying to the formal requirements of the project, some other components have been developed to give guidance [IS96c] for the assessors so that they can satisfy to the requirements of a SPICE conformant assessment [IS96b]. This should also make the assessments reliable and comparable.

SPICE Components

The SPICE deliverables is a set of 9 documents, currently with the status of Technical Reports. Some of them are intended to become *Standard*, and other will remain with the status of Guidance. (Similar approach can be found with ISO9001 standard and ISO9000-3 guidance). The document set contains :

- ◊ Part 1 : Concepts and introductory guide (informative),
- ◊ Part 2 : A reference model for processes and process capability (normative),
- ◊ Part 3 : Performing an assessment (normative),
- ◊ Part 4 : Guide to performing assessments (informative),
- ◊ Part 5 : An assessment model and indicator guidance (informative),
- ◊ Part 6 : Guide to qualification of assessors (informative),
- ◊ Part 7 : Guide for use in process improvement (informative),
- ◊ Part 8 : Guide for use in determining supplier process capability (informative),
- ◊ Part 9 : Vocabulary (informative).

The Part 2 [IS96a] describes the two dimensions of the *reference model* : the process dimension and the capability dimension with its 6 levels. It gives a reference framework for the existing assessment methods, so that their assessment results can be input in the SPICE framework to be compared together. SPICE also gives an *assessment model* [IS96d], that can be use by itself, to perform assessments. The Part 3 contains the requirements to perform a full SPICE conformant assessment. The Figure 2 shows how the reference model is embedded in the assessment model.

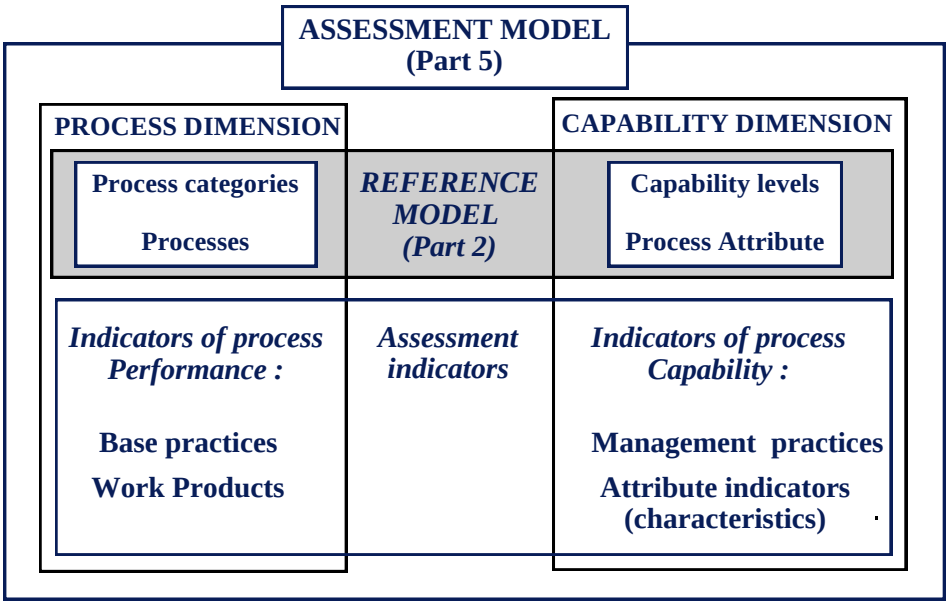


Figure 2 : The reference model and the assessment model

Assessment Model for Software Process

The two dimensions of the assessment model are described below. The Part 5 contains all the details of the *Process performance indicators* and the *Process capability indicators*, both used by the assessor to make his judgment on the achievement of a specific *Attribute* characterizing a *Level*. The process dimension is quite close to the process/activity list given by the ISO/IEC standard 12207, *Information technology - Software life cycle processes* [IS95].

The process dimension contains 5 *Process categories* :

- ◊ The *Customer-Supplier* process category consists of processes that directly impact the customer, support development and transition of the software to the customer, and provide for its correct operation and use.
  - CUS.1 Acquire software
  - CUS.2 Manage customer needs
  - CUS.3 Supply software
  - CUS.4 Operate software
  - CUS.5 Provide customer service
- ◊ The *Engineering* process category consists of the basic processes to establish a software life-cycle, in the common sense of the term and whatever it is.
  - ENG.1 Develop system requirements and design
  - ENG.2 Develop software requirements
  - ENG.3 Develop software design
  - ENG.4 Implement software design
  - ENG.5 Integrate and test software
  - ENG.6 Integrate and test system
  - ENG.7 Maintain system and software
- ◊ The *Support* process category consists of processes that may be employed at any time, and by any of the other processes of the model.
  - SUP.1 Develop documentation
  - SUP.2 Perform configuration management
  - SUP.3 Perform quality assurance
  - SUP.4 Perform work product verification
  - SUP.5 Perform work product validation
  - SUP.6 Perform joint reviews
  - SUP.7 Perform audits
  - SUP.8 Perform problem resolution
- ◊ The *Management* process category consists of processes that contain practices of a project management nature.
  - MAN.1 Manage the project
  - MAN.2 Manage quality
  - MAN.3 Manage risks
  - MAN.4 Manage subcontractors
- ◊ The *Organization* process category consists of processes that establish the business goals of the organization and develop process, product, and resource assets which, when used by the projects in the organization, will help the organization achieve its business goals.
  - ORG.1 Engineer the business
  - ORG.2 Define the process
  - ORG.3 Improve the process

ORG.4 Provide skilled human resources
ORG.5 Provide software engineering infrastructure

Every the process is defined in term of *Goal* : it means that during an assessment, the challenge for the assessment team will have, first to estimate if the process satisfy to its purpose, and then to establish the process capability by referring to the attributes of the levels.

The 6 levels of the Capability dimension are :

Level 0 *Incomplete* : There is general failure to attain the purpose of the process. There are no easily identifiable work products (document/data) or outputs of the process.

Level 1 *Performed* : The purpose of the process is generally achieved. The achievement may not be rigorously planned and tracked. Individuals within the organization recognize that an action should be performed, and there is general agreement that this action is performed as and when required. There are identifiable work products for the process, and these testify to the achievement of the purpose. The level has only the *1.1 Process performance* attribute (see Figure 2).

Level 2 *Managed* : The process delivers work products of acceptable quality within defined timescales. Performance according to specified procedures is planned and tracked. Work products conform to specified standards and requirements. This level focuses on 2 attributes : the *2.1 Performance management* attribute that adresses the performance of the process (planning and tracking of the tasks and activities) and the *2.2 Work product management* attribute (Configuration management and quality characteristics of the process outputs).

Level 3 *Established* : The process is performed and managed using a *Defined process*, this usually means a standardized process for the Organization. Individual implementations of the process use approved, tailored versions of standard, documented processes. The resources necessary to establish the process definition are also in place. The level 3 has 2 attributes : the *3.1 Process definition* attribute and the *3.2 Process resource attribute*.

Level 4 *Predictable* : The Defined process is performed consistently in practice within defined control limits, to achieve its goals. Detailed measures of performance are collected and analyzed. This leads to a quantitative understanding of process capability and an improved ability to predict performance. Performance is objectively managed. The quality of work products is quantitatively known. The level 4 has 2 attributes : the *4.1 Process measurement* attribute and the *4.2 Process control* attribute.

Level 5 *Optimizing* :Performance of the process is optimized to meet current and future business needs of the organization. The process achieves repeatability in meeting its defined business goals. Quantitative process effectiveness and efficiency goals (targets) for performance are established, based on the business goals of the organization. Continuous process monitoring against these goals is enabled by obtaining quantitative feedback and improvement is achieved by analysis of the results. Optimizing a process involves piloting innovative ideas and technologies and changing non-effective processes to meet defined goals or objectives. This level has 2 attributes : the *5.1 Process change* attribute and the *5.2 Continuous improvement* attribute

Performing the process assessment

In this chapter, we describe one possible way to perform a SPICE assessment. Some of the requirements expressed in [IS96a], as well as the tasks achieved for every steps of the assessment are illustrated below. For that purpose, the data are associated to an assessment

that was performed to experiment the new capability and process dimensions (SPICE *v2)*[IS96b]within an organizational unit developing technical software.

Even if the experiment was not dedicated to an actual process improvement or capability determination initiative, it has been decided to follow the rules in order to do a SPICE conformant assessment. These rules involve to :

- ◊ use a model compatible with the reference model : Part 5 [IS96d) has been selected,
- ◊ review defined inputs, and record and document the justification of the assessment results,
- ◊ satisfy certain requirements of the Part 2 document [IS96a],
- ◊ plan the expected outputs (process profiles, improvement orientations, etc.]
- ◊ to include in the assessment team a qualified assessor : we have acted as lead assessor during the Phase 1 Trials and for other SPICE assessments; feedbacks are described in [SI96].

Assessment Process

The assessment process is described in an assessment plan written by the qualified assessor, checked by the co assessor and approved by the Sponsor. The assessment plan contains the following items :

1. Introduction - Context
2. Terminology
3. Reference documents
4. Confidentiality agreement
5. Assessment inputs
   - *Assessment goals*
   - *Assessment scope*
     - *Process context*
     - *Selected processes*
     - *Level to be assessed*
     - *OU's units concerned*
     - *Process instances/projects*
   - *Constraints*
   - *Principles*
6. Assessment outputs
   - *Process capability profiles*
   - *Process capability levels*
   - *Assessment Report*
   - *Experiment Report*
7. Role and responsibilities
   - *Assessment Sponsor*
   - *Lead Assessor*
   - *Co Assessors*
   - *OU Facilitator*
   - *Participants*
8. Progress tracking and Quality control
9. Assessment performance
   - *Planning*
   - *Briefing and training*
   - *Assessment technic and tools*
   - *Resources - Infrastructure - Logistic*
   - *Documents inputs*
   - *Data collecting*
   - *Data validation*
   - *Debriefing*
   - *Results presentation*
   - *Next action*
10. Detailed planning

Some of these items are developed further in the chapter.

The plan is used as a road map, all along the assessment; It is also very helpful to control the progress of the experiment during some quality control actions. The document part 4 [IS96c] gives other guidance to organize an assessment and to write an assessment plan.

Assessment Organization and Planning

In the context of our experiment, the overall tasks have been planned on a 4 weeks basis, according to the planning represented in Figure 3. The assessment scope included 8 process at project level and 4 processes adressing the Organization ; 2 process instances were selected.

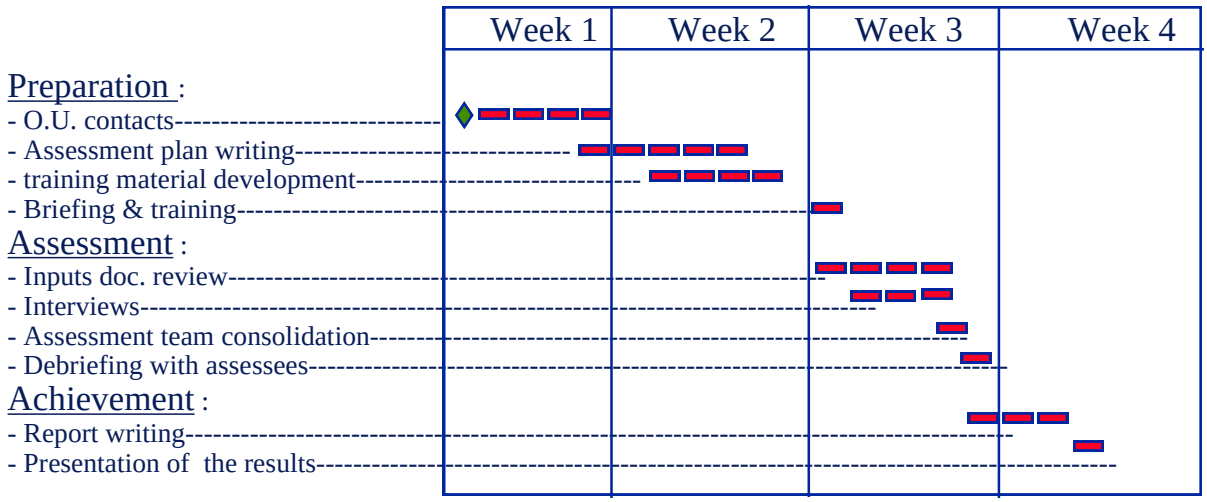


Figure 3 : Example of a typical assessment planning

This planning only gives a high level view, it should be completed in the assessment plan with a more detailed one, that indicates very precisely when (day and time) the different meeting and interviews are performed. The detailed schedule should be establish in collaboration with the *OU Facilitator* in order to guarantee the availability of each individual involved in the assessment. To define the detailed schedule, the following points must be clearly identified :

- ◊ assessment tasks to be performed (inspections, meetings, interviews, feedback sessions, etc.),
- ◊ Assessment scope, according to the assessment purpose,
- ◊ Process instances, according to the assessment scope,
- ◊ Timing allocated for each interview/meeting, according to the resources and infrastructure availability and considering the assessment budget,
- ◊ Project team and Individuals involved,
- ◊ Availability of documents (*assessment inputs*),
- ◊ Assessment Sponsor expectations.

A draft version of the detailed scheduled should be given to the OU as soon as possible to establish the final one with a collaborative approach between the assessment team and the OU.

Assessment Inputs

The assessment Team has to identify and to describe in the assessment plan all the inputs used during the assessment.

- ◊ Assessment goals/ purpose : a very clear understanding of the assessment goal as it is defined by the Sponsor, is a key issue for the success of the assessment. You may assess for process improvement for example, but your improvement strategy will certainly impacts the assessment scope. If you want to experiment the assessment technic and/or participate to the SPICE Phase 2 Trials, your approach will be different.
- ◊ Process context : the SPICE assessment method asks to the assessment team to consider the process environment in order to make its judgment and determine the process attributes ratings and process capability level. Some specific processes, let's consider for example *ENG.3 Develop Software Design* or *MAN.3 Manage Risks* might have different requirements if the process is instanciated for a real-time embedded system or for a client-server application lasting only 3 man/month.
- ◊ Level to be assessed : indicates if the investigation covers the 6 capability levels or only a subset of it.
- ◊ Selected processes : The processes to be assessed will be identified according to the inputs described up above See also [SI95]. A mapping between the standard SPICE processes and the process of the OU might be necessary in case of specificities within the OU. This mapping might be the occasion to develop some *extended process*, that are not existing in the standard model [IS96a].
- ◊ Process instances/projects : The selected processes have to be assessed with *process instances*, that is at the level of the project and/or the organization (units, department, teams, etc.).
- ◊ Constraints : Any constraint that has to be considered during the assessment has to be identify, for example : resources availability, confidentiality issues, Sponsor requirements, assessment output data collection, etc.
- ◊ Principles : this part should briefly explain what is the SPICE assessment philosophy for those (Executives, etc.) who will received the Assessment plan and may not attend the training sessions, interviews or debriefings.
- ◊ Additional information to be collected : the assessment by the mean of the interviews (if this basic technic is used for the assessment) give the opportunity to collect some additional data like : improvement actions to be conducted, identified difficulties in performing project tasks or to implement internal standard/procedures, etc.

Responsibilities

The role and responsibilities of the participants involved in the assessment process are described in the assessment plan. See [IS96c]. Some specific issues are described there :

- ◊ Assessment Sponsor : He can strongly influence the attitude of the assessment participants (the assessees) so that they have (or not) a positive and collaborative approach during the interviews.

- ◊ Lead Assessor : The assessment success depends on his knowledge of the SPICE framework and assessment technics.
- ◊ Co Assessors : They must help the Lead Assessor with their expertise for some processes and should be able to establish their own judgment and ratings to compare and validate with the Lead Assessor's one.
- ◊ OU Facilitator : He might be very useful when the assessment team does not know the business sector of the assessed OU. In some case, he acts as a moderator if conflictual situation occurs during an interview with some participants.
- ◊ Participants : All of them have to be briefed about the assessment before the interviews.

Assessment Team preparation

A minimum of two people is needed for the assessment team : A single Lead Assessor might feel very uncomfortable if he has to manage the interview, listen to and record the answers and proofs of conformance, prepare the next question and have a look to any SPICE documents.

The assessment team must be cohesive, knowing the parts of the SPICE model within the assessment scope (rating scheme, concept of achievement of attributes, etc.). All members of the team must be aware of he assessment : purpose, scope, constraints, approach, planning and schedule, etc. If possible, the assessment team members have to be involved in the assessment plan writing and/or verification.

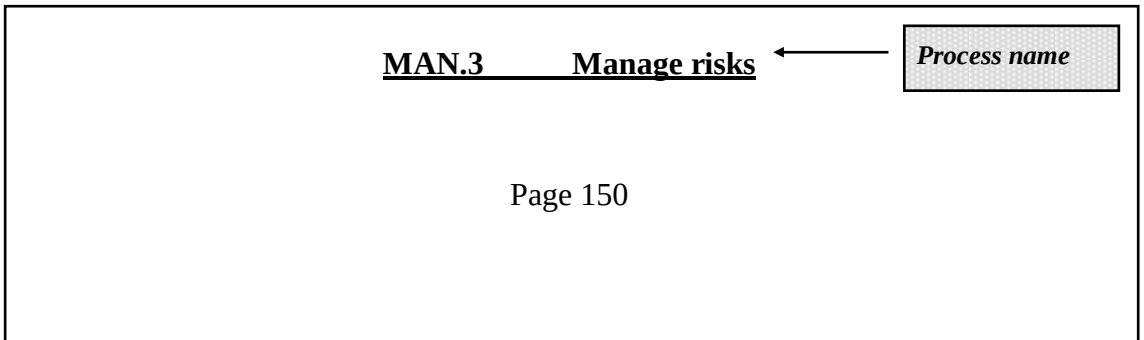
Assessment Technics

The assessment might be based on one of the following technics : interviews, individual discussions, group discussions, closed team sessions (assessment team discusses findings amongst themselves), documentation inspections, feedback sessions (assessment team discusses findings with OU representatives) or questionnaires. This will be decided between the Assessment sponsor and the Lead assessor, by considering : the assessment goal, the availability of tools, people, time and resources. A simply way to practice an assessment is to use a paper based set of the original SPICE documentation, to build some forms to record the assessment outputs (see Figure 4) and for presenting the results.

Preparing the Organizational Unit

The preparation of the Organizational Unit 's members is also very important. Basically, at least, two meetings should be organized :

- ◊ Meeting to present the assessment (+/- 1 hour) , conducted by the Sponsor and the Lead Assessor. They both present to the OU staff the assessment purpose, scope, constraints. How the assessment will be conducted. What are the benefits of assessment results and the principles for confidentiality and ownership.
- ◊ A training session for the assessment participants - the interviewed (A half day is a minimum). Members of the assessment team explains the SPICE assessment approach and method.



The purpose of the Manage risks process is to continuously identify and mitigate the project risks throughout the life cycle of a project. The process involves establishing a focus on management of risks at both the project and organizational levels. As a result of successful implementation of the process:

◊ the scope of the risk management to be performed for the project will be determined;

◊ .....

◊ corrective action will be taken when expected progress is not achieved.

Process goal description

Base practice (= Process performance indicator)

MAN.3.1 Establish risk management scope. Determine the scope of risk management to be performed for this project.

Note: Issues to be considered include the severity, probability, and type of risks to identify and manage.

Assessment data

Existence : No ☐ Yes ☑ Adequacy : Not ☐ Partially ☐ Largely ☑ Fully ☐

Notes : Le champ de la gestion des risques concerne au niveau RSO, les risques de type coûts, délai et qualité associés à la réalisation de prestations forfaitaires. RSO a établi une méthodologie nommée MARS, pour l'analyse et la diminution des risques. Cette dernière est en cours d'application expérimentale sur un projet.
Proof of conformance : Voir le Document Réf

Base practice

MAN.3.2 Identify risks. Identify risks to the project as they develop.

Note: Risks include cost, schedule, effort, resource, and technical risks.

Existence : No ☐ Yes ☑ Adequacy : Not ☐ Partially ☐ Largely ☐ Fully ☑

Notes : Réalisé, dans le champ défini en MAN3.1.
Proof of conformance :

Figure 4 : Form to collect assessment outputs (extract)

Potential problems

When preparing the assessment, a risk analysis should be done to guarantee the achievement of the assessment goals. Among potential risks are : unavailability of documentation, organizational Unit 's individuals availability, resistance from the Organizational Unit to provide information, changes to purpose or scope of the assessment, lack of confidentiality, etc.

Feedbacks from Performing the Assessments

Process attribute ratings

The process quotation is made by the Assessment team, using *indicators* (see Figure 2). Two kinds of *Process performance indicators* exist; they are :

- ◊ *Base practices* : corresponding to software engineering or management activities that address the purpose of a particular process (a process has between 3 and 12 base practices).
- ◊ *Work products* : process inputs and outputs (data/document).

The *Process capability indicators* are :

- ◊ *management practice* : a management activity or task that addresses the implementation or institutionalization of a specified process attribute
- ◊ *management practice characteristics* : objective attributes or characteristics dedicated to the management practice performance, resource and infrastructure that support the judgment of the Assessment team of the extent of achievement of a specified process attribute.

The process performance estimations as well as the process capability evaluations are made using a specific *rating scale*.

For each assessed process instances, the first step is to estimate the base practice *existence* by using the following rating scale :

- ◊ *Non-Existent* : The base practice is either not implemented or does not produce any identifiable work products,
- ◊ *Existent* : The implemented base practice produces identifiable work products.

Then, the Base practice adequacy is evaluated by the Assessment team, by using a four levels adequacy rating scale (see Figure 5, Level 1 quotation). The second step when assessing a process instance is to estimate the adequacy of the Management practices, using the Management practice adequacy rating scale

Management practice adequacy are rated using the management practice adequacy rating scale defined below.

- ◊ *Not adequate* : The management practice is either not implemented or does not to any degree satisfy its purpose,
- ◊ *Partially adequate*: The implemented management practice does little to satisfy its purpose,
- ◊ *Largely adequate* : The implemented management practice largely satisfies its purpose,
- ◊ *Fully adequate* : The implemented management practice fully satisfies its purpose.

These scale are very practical for the Assessors to make their judgments. Instead of only answering to some Yes/No questions, the Assessment team has to consider the assessment context to estimate the adequacies to the requirements of the SPICE Model. This involve all the member of the team to have a good knowledge of the SPICE process model, and also to have a fair understanding of the quotation principles.

LEVEL 1 : This quotation is the result of the Process performance evaluation, according to the Process's Base practices, using the adequacy scale :

- ◊ ***N**ot adequate* : The base practice is either not implemented or does not to any degree contribute to satisfying the process purpose,
- ◊ ***P**artially adequate* : The implemented base practice does little to contribute to satisfying the process purpose,
- ◊ ***L**argely adequate* : The implemented base practice largely contributes to satisfying the process purpose,
- ◊ ***F**ully adequate* : The implemented base practice fully contributes to satisfying the process purpose.

Lev.	Processus Attribute	Management Practices	Result. (NPLF)	Quot. (NPLF)
1	1.1 Process performance	*Ensure base practice performance*	/	**F**
2	2.1 Performance management	*Design/Document the project plan*	L	**L**
		Verify process compliance to the plans	L	
	2.2 Work product management	*Review/Audit work products for integrity*	L	**P**
		Review/Audit work products for quality	N	
3	3.1 Process definition	*Standardize the process*	L	**L**
		Tailor the standard process	L	
	3.2 Process resource	*Allocate skilled human resources*	P	**L**
		Provide process infrastructure	F	
4	4.1 Process measurement	*Define process & product metrics*	P	**P**
		Track with measurement	P	
	4.2 Process control	*Analyze metrics & deviations*	P	**P**
		Ensure that corrective action is taken	P	
5	5.1 Process change	*Eliminate defect causes*	N	**N**
		Improve the defined process	N	
	5.2 Continuous improvement	*Establish improvement product quality goals*	N	**N**
		Establish improvement process effectiveness goals	N	
	Capability level :			**1**

Figure 5 : Form for quotations' record purpose

LEVEL 2-5 : These quotations are the results of the Process capability evaluation

The elementary quotations, obtained for each process instances has shown in Figure 5 of a specific process, are then combined together to represent the Process capability with, for example, the distribution of the attributs' cotation, among all the process instances (Figure 6).

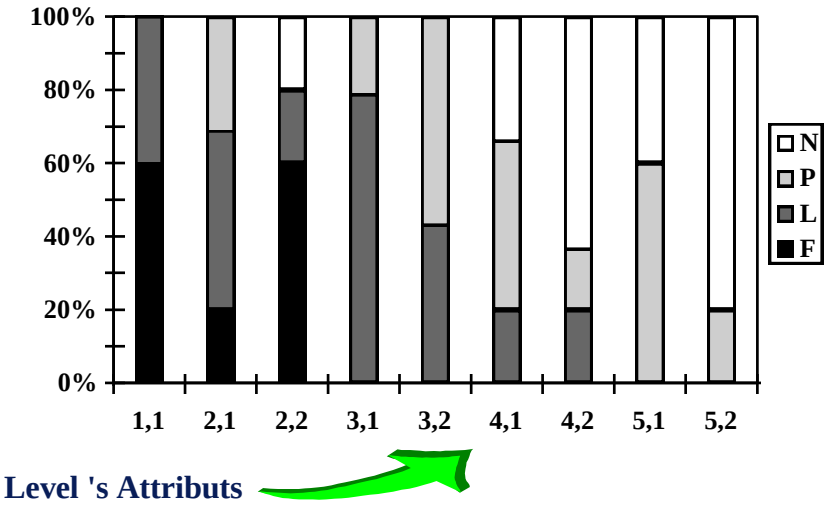


Figure 6 : Distribution of the Attributes' quotations

Other representations might be used, but one must remenber that these graphical results are only based on assessment recording and don't correspond to any absolute measurements (this means that any variation of +/- 10 % must not be considered significantly!).

Assessment report

The graphical representation of Figure 6, as one of the assessment outputs, is useful to have an global overview on a process capability. Nevertheless, it is essential to write a full *Assessment report*, to record all the collected data. According to the assessment goal, some other documents might be establish as an Improvement Plan, and/or an Assessment experiment report. Below is an example of assessment report that includes somme comments for each process instances (typically improvement opportunities...) as well as some feedbacks about the assessment process itself.

1.Introduction - Context
2.Terminology
3.Reference documents
4.Confidentiality agreement
5.Assessment inputs overview
  Assessment goals
  Assessment scope
    Process context
    Selected processes
6.Assessment results
  Process capability profiles
  Process capability levels
7.Comments per process instances
  Process instances from project 1
  .....
  Process instances from project n
  Process instances from Organization 1
  ....
  Process instances from Organization n

8.Assessment feedback
Resources - Infrastructure - Logistic
Assessment technic and performance
Meeting - Training - Presentation
Planning - Schedule
9.Further actions
10.Quality control records
11.Annexes
A1 : Assessment records
A2 : SPICE process dimension
A3 : SPICE capability dimension

Conclusion

The ISO/SPICE software process model gives a reference framework for software process assessment. The framework provides a reference model, that can be used by any assessment method allowing assessments comparisons, as well as an assessment model and some guidance to perform assessments. This environment has been successfully used and feedback, like those reported in this paper, are getting available for the Software Community.

The guidance for process improvement and capability determination are strategical and the practical issues given by ISO/SPICE should motivate anyone involved in software as a user, a customer or a supplier. Many of them should be attracted to be involved in the SPICE Phase 2 Trials which have recently started.

References

[EL96] El Eman K., *Some initial results from the international SPICE trials*, Software Process Newsletter, n°6, Spring 1996.

[IE 97] *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, to be published by IEEE CS Press in 1997.

[IS95] ISO/IEC 12207, *Information technology - Software life cycle processes*, first edition Aug. 1995.

[IS96a] *Software process assessment - Part 2 : A reference for process and process capability*, WD N102, ISO/IEC/JTC1/SC7/WG10, July 1996.

[IS96b] *Software process assessment - Part 3 : Peforming an assessment*, WD N103, ISO/IEC/JTC1/SC7/WG10, July 1996.

[IS96c] *Software process assessment - Part 4 : Guide to performing an assessment*, WD N104, ISO/IEC/JTC1/SC7/WG10, July 1996.

[IS96d] *Software process assessment - Part 5 : An assessment model and indicator guidance*, WD N103 version 1.06, ISO/IEC/JTC1/SC7/WG10/OWG, 18 Aug. 1996.

[MA96] Marshall P., Maclennan F and Tobin M., *Analysis of observation and problem reports from Phase 1 of the SPICE trials*, Software Process Newsletter, n°6, Spring 1996.

[SI95] Simon J-M, *Choosing the process for the assessment scope : a key factor for SPICE assessment success*, Proceedings of the European Conference on Software Process Improvement - SPI95 - Barcelona / Spain, Nov-Dec 1995.

[SI96] Simon J-M, *Experiences in the Phase 1 SPICE Trial*s, Software Process Newsletter, n°8, Winter 1996.

[SP95] *Phase 1 Trials Report*, version 1.00, Oct 1995. Spice document.

[WO96] Woodman I. and Hunter R., *Analysis of assessment data from Phase 1 of the SPICE trials*, Software Process Newsletter, n°6, Spring 1996.

Practical Experience of the Establishment of Improvement Plans Using the Bootstrap Process Model

Richard Messnarz, ISCN Ltd., Dublin, Ireland
iscn@genrix.ie
http://www.iol.ie/~iscn/homepages/bootstrap/index.html

Abstract:

Many publications about BOOTSTRAP deal with the assessment process, the underlying BOOTSTRAP process architecture, and the evaluation approach calculating strengths and weaknesses profiles of organisations [4], [6], [7], [8], [10], [11]. The author has extensively performed research over the last 5 years continuously working on the refinement of the BOOTSTRAP assessment process.
However, as outlined by well known organisations the effort for assessment forms only 3-5% of the overall improvement effort needed. Therefore any assessment is only a starting point and the quality of the action plan as the result of the assessment is of critical importance for the success or failure of the improvement initiative. It is not enough to do an assessment and it is the remaining 95-97 % of the improvement effort to achieve top management commitment, to motivate practitioners for doing the improvement actions, for actually carrying out the improvement projects, and for measuring the success or failure [3], [9], [17].
This article specifically discusses the author's experience with the use of the BOOTSTRAP assessment process and maturity profiles for establishing business based and goal driven action plans which ensure a successful performance as much as possible. The article contains case studies from different organisations to illustrate different approaches because one and the same situation in different environments requires different (and adapted) actions.

Introduction

The introduction gives a short overview of BOOTSTRAP´s current version [7] and illustrates typical results of BOOTSTRAP assessments. The strengths and weaknesses profiles are used to establish a list of recommended actions and to prioritise them. According to the author´s experience the definition of such priorities is a very important factor and section 3. *Different Approaches for Defining Priorities* deals with this aspect in four case studies. Section *4. Framework for Detailed Improvement Planning* discusses an approach which uses priorities (discussed in the previous section) and establishes a set of improvement projects compliant with the standard BOOTSTRAP action plan structure.

Each software process assessment and improvement model has two dimensions that are *functionality* and *capability***.** The ***functionality dimension*** contains the processes to be evaluated and the ***capability dimension*** the capability or maturity levels against which the processes will be evaluated.
The ***functionality dimension*** of the BOOTSTRAP Process Architecture Version 2.3 was formed enhancing the processes of the SEI model (formed according to DoD 2167A standard (DOD-STD-2167A, 1988) with processes aligned with the ISO 9001 and ISO 9000-3

requirements, the processes of the ESA life-cycle model, and general TQM principles. This resulted in a tree based architecture of process clusters and processes as outlined in Figure 1.

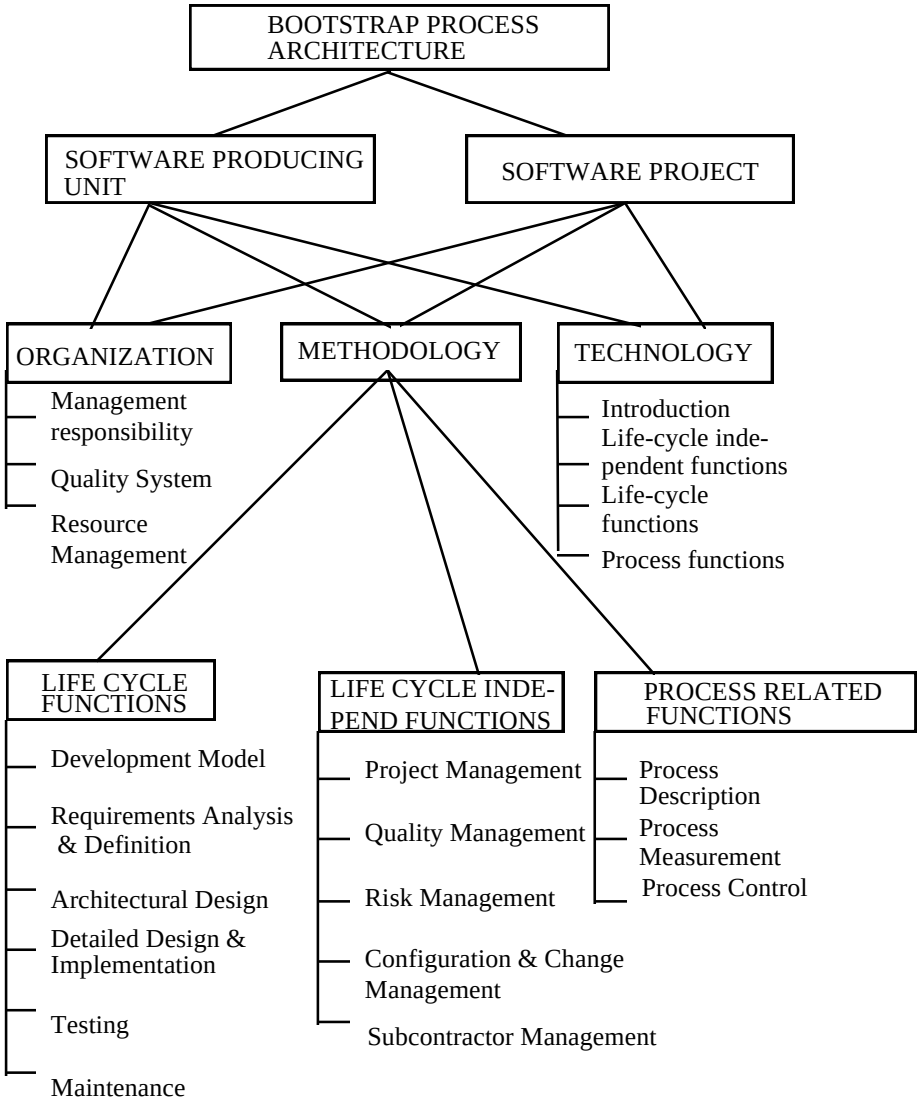


Figure 1. The BOOTSTRAP process model version 2.3 - the functionality dimension

The BOOTSTRAP process architecture of version 2.3 contains process clusters (e.g. methodology, life cycle functions), each process cluster contains a number of process attributes (e.g. project management in the process cluster life cycle independent functions) and each process attribute contains a number of questions (checkpoints) which are evaluated on a 4 point linguistic scale (absent, basic, significant, extensive). Questions can also be answered as Not Applicable.

Meanwhile a SPICE [2], 13] compliant version 3 of the BOOTSTRAP process architecture is field tested in the SPICE phase 2 trials. This architecture is not published so far and takes the following aspects into account.

BOOTSTRAP version 3 bases on the ISO 12207 (see Figure 2) life cycle processes standard which formed the basis for the SPICE architecture. This process standard contains three process categories (comparable to the process clusters in version 2.3) consisting of a number of processes (comparable to the process attributes in version 2.3) each of which contains a checklist if practices are performed (comparable to the list of base practices defined in SPICE).

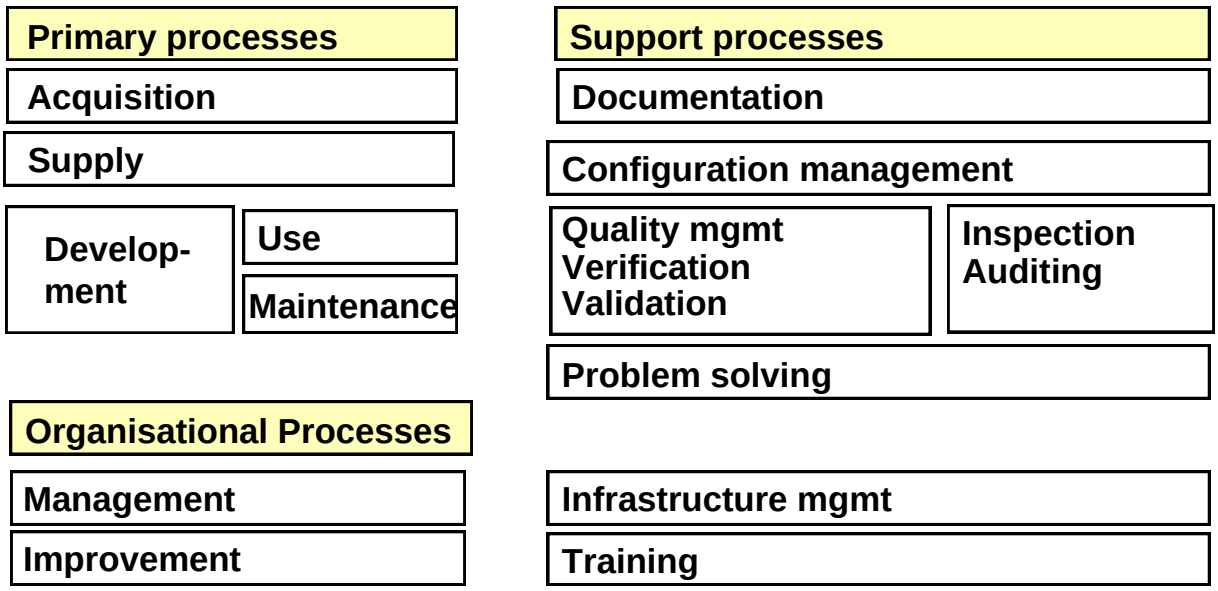


Figure 2. ISO 12207 Life Cycle Processes

The capability dimension of the SPICE process model defines evolving capability of the implemented or institutionalised processes in terms of capability levels[16], process attributes (former common features)[17], and generic practices[18]. The decomposition was derived based on grouping by type of implementation or institutionalised activity
There are six capability levels in the SPICE process model, that are: Incomplete (0), Performed (1), Managed (2), Established (3), Predictable (4), and Optimising (5).

In the SPICE capability dimension each process of the functionality dimension may have features in its implementation that address on each of the capability levels. In the BOOTSTRAP process model each base practice is assigned to one capability level (see dotted lines in Figure 3), but a process may include base practices from 2 to 3 different capability levels and may, therefore, span over many capability levels which enables the BOOTSTRAP methodology to evaluate each process separately on the capability scale, providing the basis for the calculation of capability profiles [3], [4], [7] (see Figure 4).
In SPICE first it is checked if the base practices are performed, then for those which are performed it is checked whether they are planned and tracked and therefore managed, and for those which are managed it is checked whether they are performed according to a defined process and therefore established, etc. And for each of these evaluations against a capability level (type of implementation) adequacy vectors (see Figure 5) are calculated based on the fact that each checkpoint is answered on the four point linguistic scale: Fully Adequate, Largely Adequate, Partially Adequate, and Not Adequate.

[16] A capability level is a set of process attributes (former common features) that work together to provide a major enhancement in the capability to perform a process.

[17] A process attribute (former common feature) is a set of practices that address an aspect of process implementation or institutionalisation.

[18] A generic practice is an implementation or institutionalisation practice that enhances the capability to perform any process.

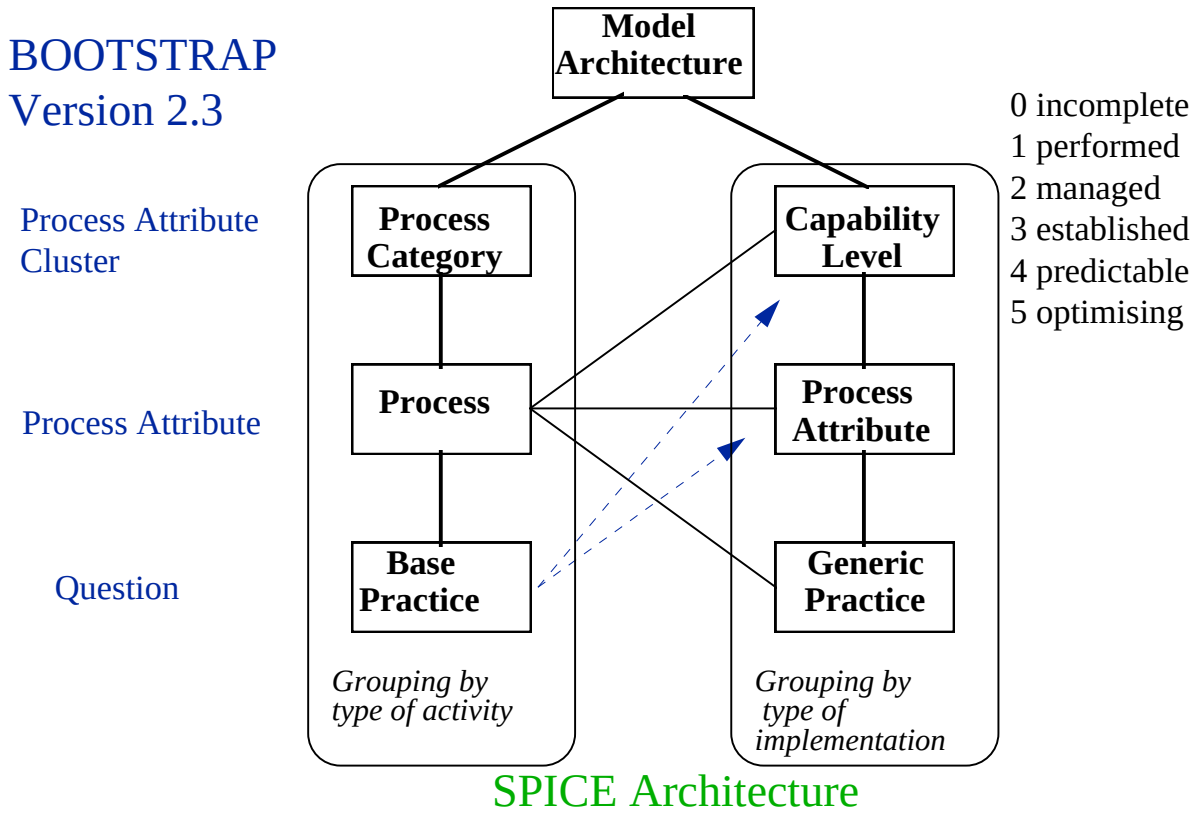


Figure 3. The SPICE process model compared to BOOTSTRAP version 2.3 [7]

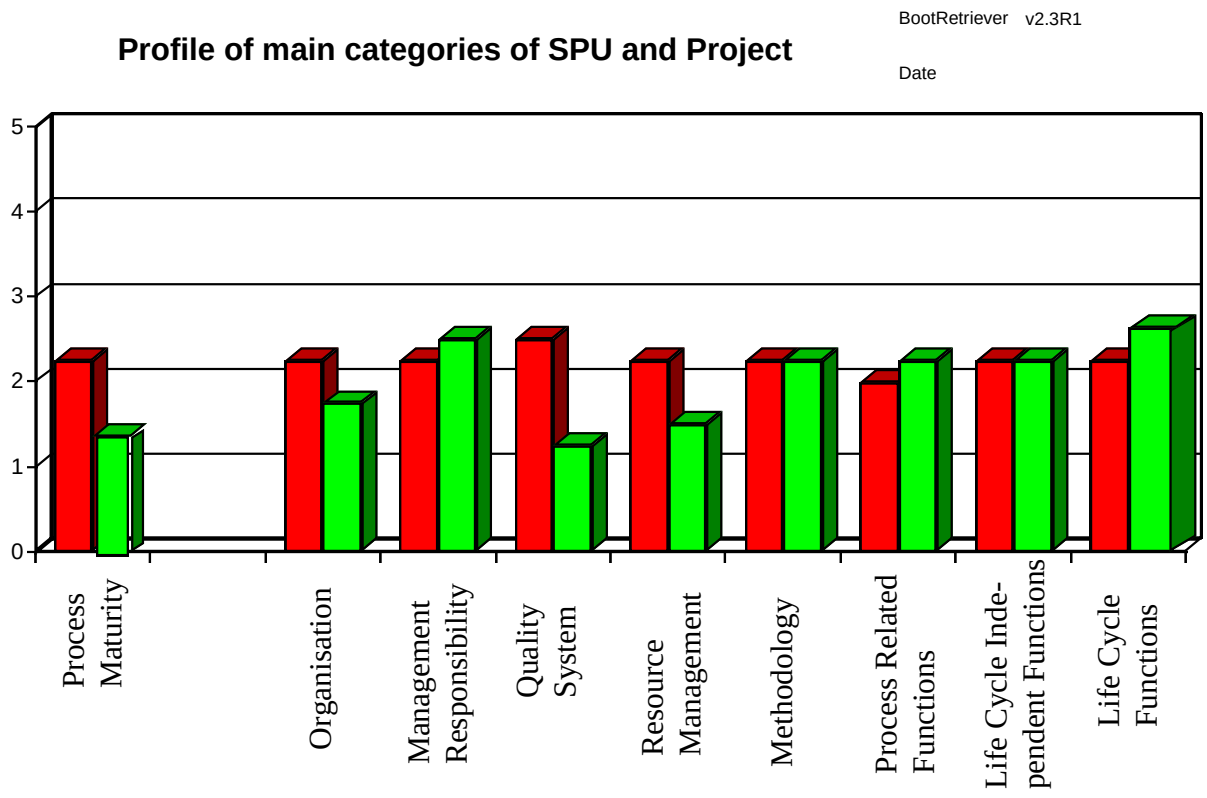


Figure 4. Typical BOOTSTRAP Maturity Level Profile

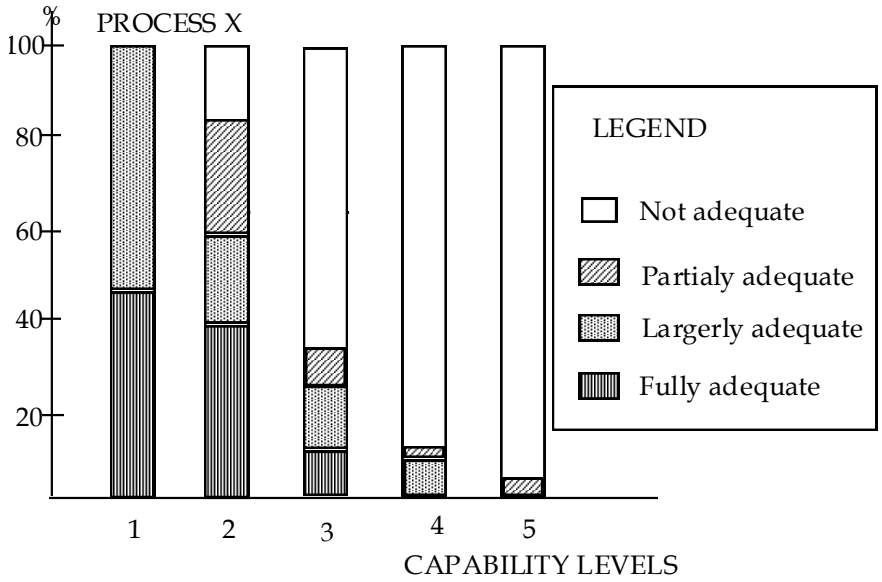


Figure 5. Typical SPICE Adequacy Profile for Process X

In future BOOTSTRAP version 3 will provide both profiles: capability level profiles calculating a maturity level per process, an adequacy profile per process explaining in more detail (with adequacy vectors) why this maturity level was calculated (see Figure 6). This way BOOTSTRAP keeps its major feature and becomes SPICE compliant [7].

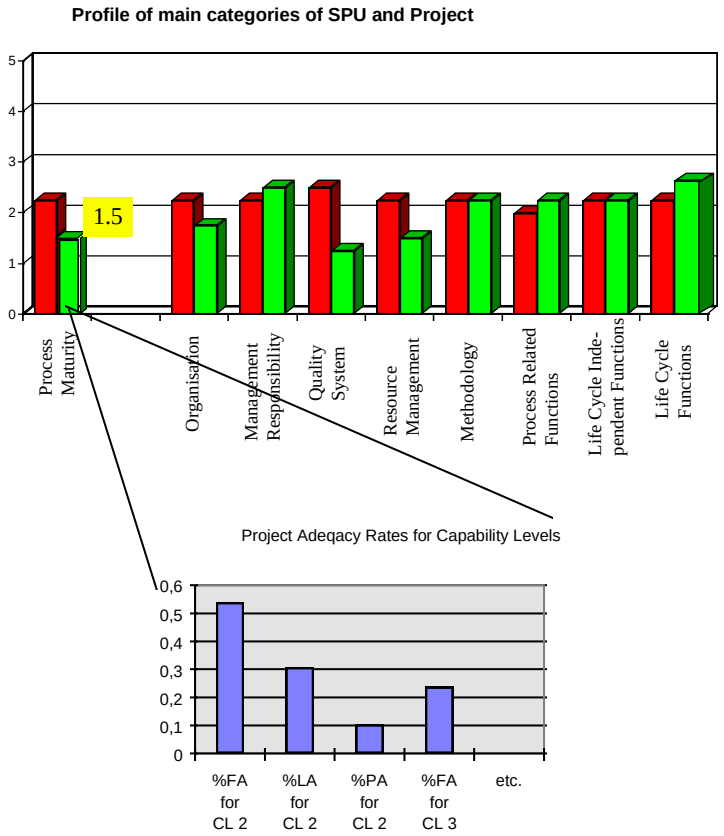


Figure 6. Combination of Maturity Profiles with Detailed Adequacy Rates

Process Improvement Planning

The SPICE PIG (Process Improvement Guide, Figure 7) describes procedures and necessary steps to plan improvement taking into account

1. the capability profiles as assessment output
2. typical profiles of industry in the same sector (benchmarking)
3. the practices of the BPG as a guideline about which practices should be in place
4. target profiles describing the improvement goals in terms of capability profiles to be achieved
5. and improvement plans including a measurement plan for collecting records to analyse cost benefit, return on investment, success or failure

to finally achieve the improvements in the organisational unit's software process.

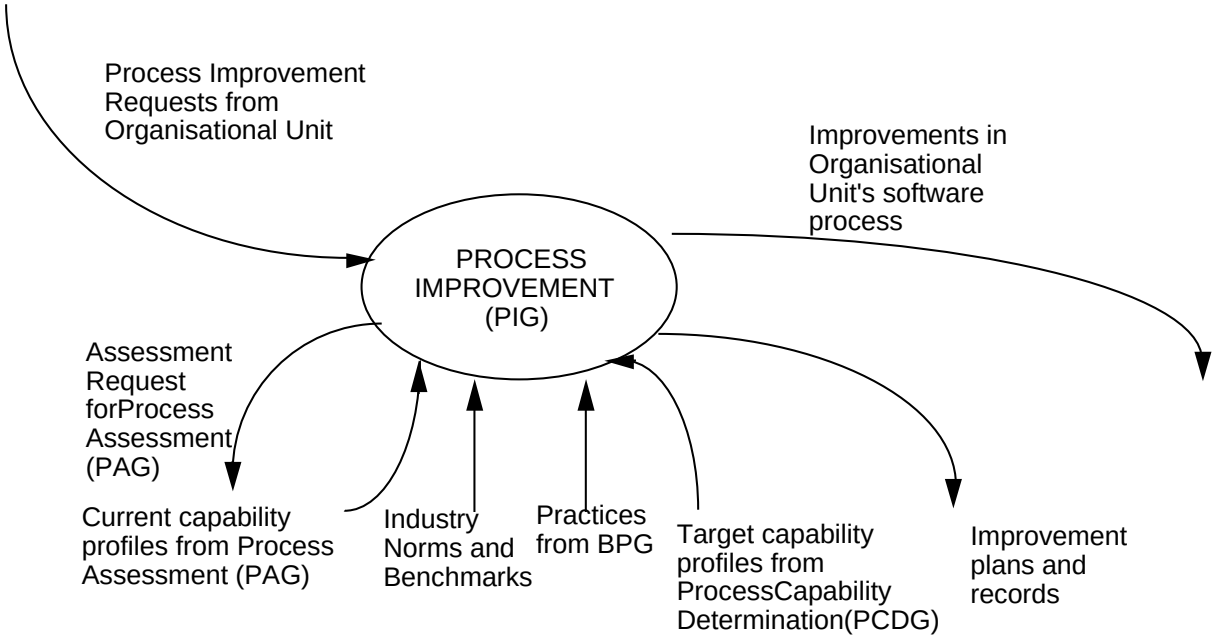


Fig. 7: Factors to Have a SPICE Compliant Improvement Planning Process

BOOTSTRAP version 2.3 largely covered most of these topics by

1. providing BOOTSTRAP maturity profiles as assessment output
2. running a European wide BOOTSTRAP database supporting benchmarking
3. providing a BOOTSTRAP process architecture with a set of practices
4. providing a standard template for assessment reports and action plans including the definition of business goals, priorities, and target profiles
5. including in the template for action plans a framework for improvement projects which requires to define measurable goals to identify the ROI and if and how the goals have been achieved.

BOOTSTRAP version 3 will be fully SPICE compliant and will (as outlined in section 1)

- cover all base practices from the BPG (Baseline Practices Guide)
- use the 6 capability levels defined in SPICE
- calculate adequacy vectors for each process in addition to the maturity level profiles.

How to define Priorities

This section contains 4 case studies from different organisations which performed BOOTSTRAP assessments and used different approaches for priority definition. This section outlines the different approaches, the underlying strategy, and why the presented approach was important to be followed in a certain organisational environment.

Business Need Driven Model

This case study is based on a small software development unit of a middle sized company in which software only forms a small part of the business. However, it was expected that in future the HW products will increasingly contain software control functionality. The major problem in the division which formed the reason why they ordered the BOOTSTRAP assessment was the fact that already 70% of the personnel was doing maintenance and the maintenance effort was still increasing so that sources for doing new development and exploiting new market potentials were blocked.
The assessment results (see Fig. 8) illustrated that the maturity of the early life cycle phases was very low and only the maintenance processes were appropriately defined. When identifying weak points in the profile BOOTSTRAP assessors look at the specific answers in the questionnaire and make a more detailed factor analysis identifying a number of factors why the maturity is low for certain processes. Figure 9 outlines the reason for the weaknesses visible from Figure 8 and illustrates a sample representation used by the author during assessments.

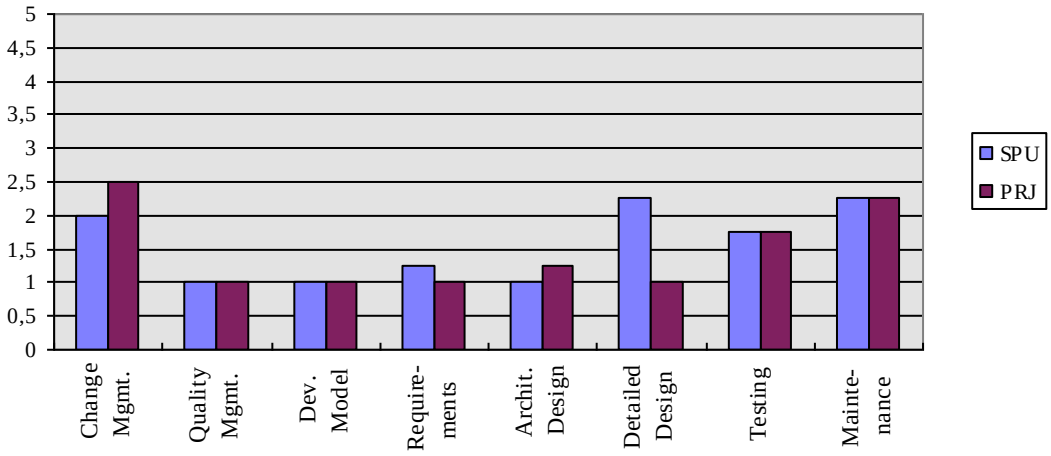


Figure 8: Sample Part of the capability Profile of case Study 1 (1992)

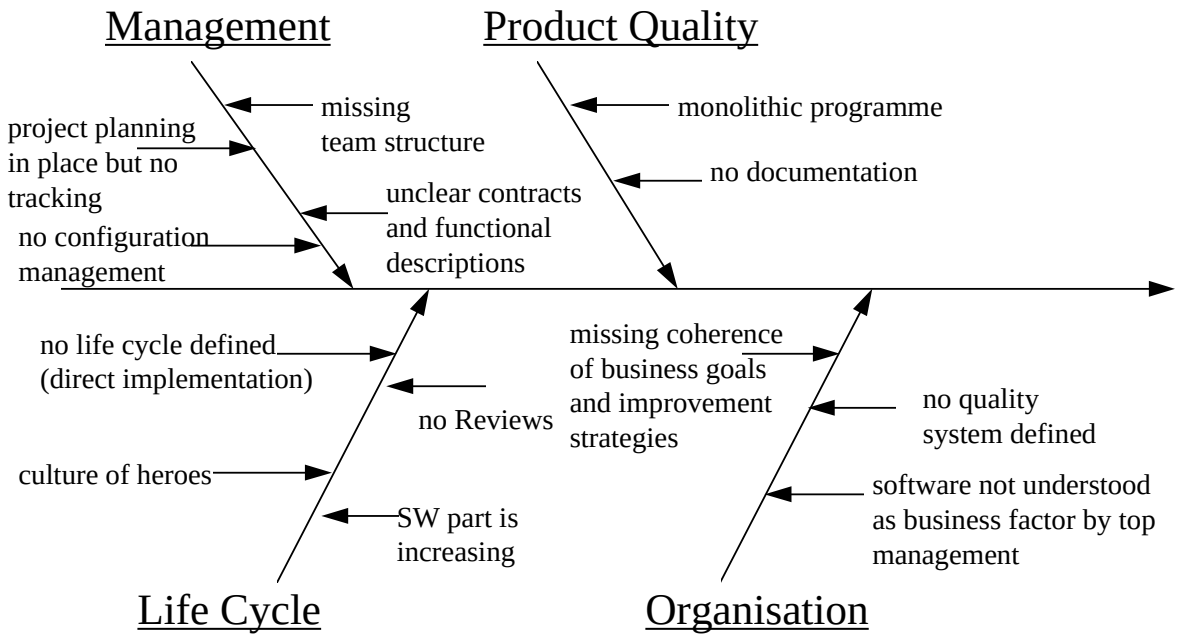


Figure 9: Analysing the Process Factors Causing the Low Maturity Levels

All software developers in the division originally were hardware engineers and started to develop software without using a life cycle, specification, or design. This did not create a problem as long as the programme which they developed (and needed in the embedded systems) was quite small. However, during 5 years the size of the software grew and due to missing architectural design the product had a monolithic structure. If you changed parts of the system it was not visible which and how many other parts were affected. Based on this situation the continuously incoming customer wishes and maintenance activities nearly always caused a number of additional not expected change activities. All parts of the system were so dependent on each other that any change led to a re-compilation of the entire product which at this time consisted of over 150000 lines of code.
Another major problem (based on a monolithic team culture) was that all people were responsible for all parts of the system which caused team problems. However, an assignment to different functions was not possible due to the missing modular architecture and definition of interfaces.

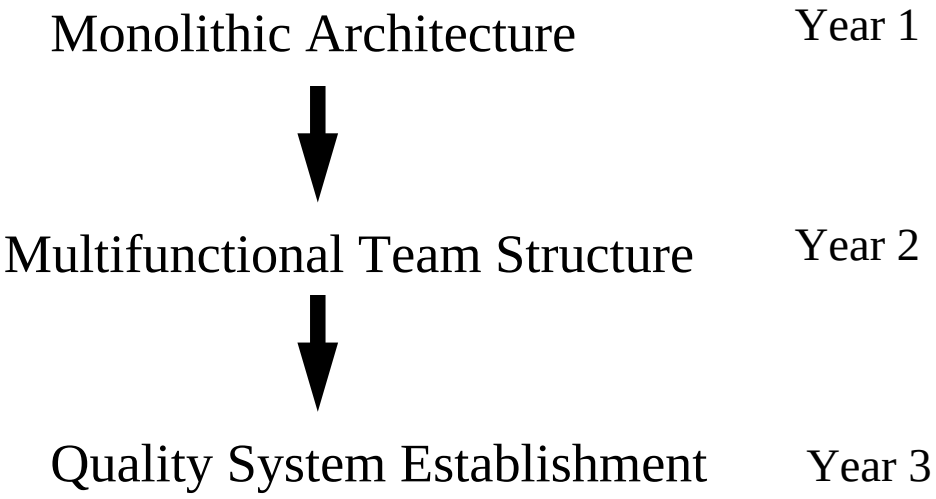


Figure 10: Definition of Priorities by Business Needs

As shown in Figure 10 therefore three major phases for improvement were defined with first solving the monolithic problem and introducing missing development techniques, secondly introducing a team model clearly defining roles, responsibilities, workflows and management tasks, and thirdly a quality system has been established and maintained which was ISO 9001 certified in 1996.

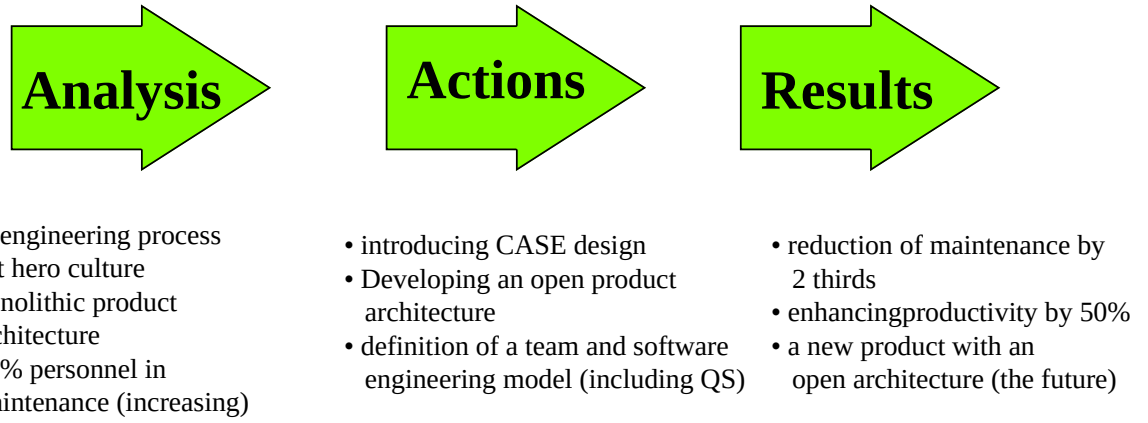


Figure 11: The Achievements

A major achievement beside productivity and quality improvements (see Figure 11) was the development of a new product based on an open architecture to which a number of components can be integrated via standardised interfaces. This led to an optimum configurable system as a future product on the market [16]. The first assessment rated the company on maturity level 1.25 and after 3,5 years improvement work (with the open architecture approach, the team model, the defined quality system) a self assessment roughly estimated a maturity level between 2,5 and 3. The company was satisfied with achieving ISO 9001 and was so far reluctant in doing a BOOTSTRAP re-assessment.

In this case study there was absolutely no choice for defining the priorities because the business needs overruled any free choice.

People Centred Model

Case study 2 discusses a division of a large company which develops embedded systems integrating electronics equipment, processors, software, and mechanics. At the time of performing the BOOTSTRAP assessment the size was about 70 people. The teams in this division have an interdisciplinary structure: e.g. 1 team leader, 2 software developers, 1 electronics expert, 1 mechanics expert, etc. A major problem in this interdisciplinary culture is to define system architectures in a way that all people (with completely different background) understand their duties and tasks and interfaces to the other team members.
Due to the organisation's success on the market and due to the feeling of all people that they were on-top of research and development the managers as well as the engineers (of different disciplines) were very self confident.

However, the following problems motivated the organisation to do a BOOTSTRAP assessment. The main customer demanded a maturity level 3 and the organisation had to determine its capability to identify the necessary steps and a reliable time frame to achieve the required maturity. The management had started to introduce ISO 9001 and development guidelines but overloaded e.g. the software developers with work so that time and resource problems occurred. The embedded systems were used in a manufacturing process to be integrated into larger systems (due to confidentiality it is not possible to mention the type of system) produced at high numbers. If all other suppliers only had to wait for the delivery of this product this caused incredible amounts of penalty payments so that time was the most important requirement. Therefore situations happened were quality reviews were defined but the management decided not to perform them to keep the time deadline. So the following people management problem arose. The managers defined more and more standards and guidelines and (without restructuring the resources and providing additional resources) asked the engineers to do more and more. And the resource restrictions led to the non-performance of required and defined quality actions as well as a cultural problem. When performing the assessment it was of critical importance to build a bridge between management and the engineers and to identify improvement actions supported by top management and showing a high motivation of the engineers to really implement it. Therefore the following „moderator model" as described below was applied.

Table 1 shows the result of the improvement meeting applying the *moderator model*.

Improvement Area	PRJ ML	ORG ML	PRI Manager	PRI Engineer	Logical Sequence	Team Assignment
Testing	1,25	1,75		18	1	- names - effort - time frame
Development Model	1	2,25	4	16	1	
Architectural Design	1,75	1,75		10	1	
Re-Use Concept			2	9	3	
Configuration and Change Management	2,25	2,25		6	2	
Programming by Contract				1	2	
Reviews	2,25	2,25	2	8	1	
Work Place	1,25	2,25		2	1	
Training Plan	1,25	2,25		9	1	
Process Models	1,75	2,25	1	3	3	

Table 1: Sample Result of a Moderator Model Based Improvement Meeting

After calculating the BOOTSTRAP maturity profiles and establishing a first action plan an improvement meeting was performed on-site inviting all site managers, project managers, and engineers. The people were not forced to participate but finally the room was filled with 27 people. The improvement meeting started with a presentation of the assessment results illustrating the strengths and weaknesses and the key problems. Then we established a pin-

board (a very large one of course) with all proposed improvement actions listed. Each of the top managers got three red pins and could select three improvement actions on the board (see the Column PRI Managers in Table 1). Each of the project managers and engineers got three green pins and could select three improvement topics (see PRI Engineers Column in Table 1).
In addition to that we established a logical sequence of improvement activities. To establish a pool of re-usable programmes, for instance, requires that all modules are well designed, reviewed, and tested, and controlled by configuration management before they are inserted into the re-use pool. This way we established numbers like 1,2,3 where 1 means that this activity must be done before starting with an activity of „logical sequence type" 2.
All this information was then used to sort all improvement actions by priorities. Higher priority meant that

- top managers will provide resources immediately
- there is a high motivation to implement the improvement actions by the project managers and engineers
- the risk to get the same cultural problems as before are minimised (a kind of mitigation strategy).

Procedure:

```
For „logical sequence type" = 1..3 do
{

  For  PRI Engineers Max  Down to Minimum do

   {

     For PRI Managers Max Down to Minimum do  Print the Improvement Action;
    }
}
```

This procedure led to:

1. Testing
2. Development Model
3. Architectural Design
4. Training Plan
5. Reviews
6. Work Place
7. Configuration and Change Management
8. Programming by Contract
9. Re-Use Concept
10. Process Models

Pragmatic Model

Case study 3 discusses a pragmatic procedure model for defining priorities. This model bases on the fact that capability levels simply represent priorities [12], [14], [15]. Maturity level 2

practices must be performed before employing level 3 aspects, and level 3 practices must be performed before starting with level 4 processes. If using the BOOTSTRAP maturity level profiles (see Figure 12) another factor, the improvement potential, can be taken into account [9], [17].

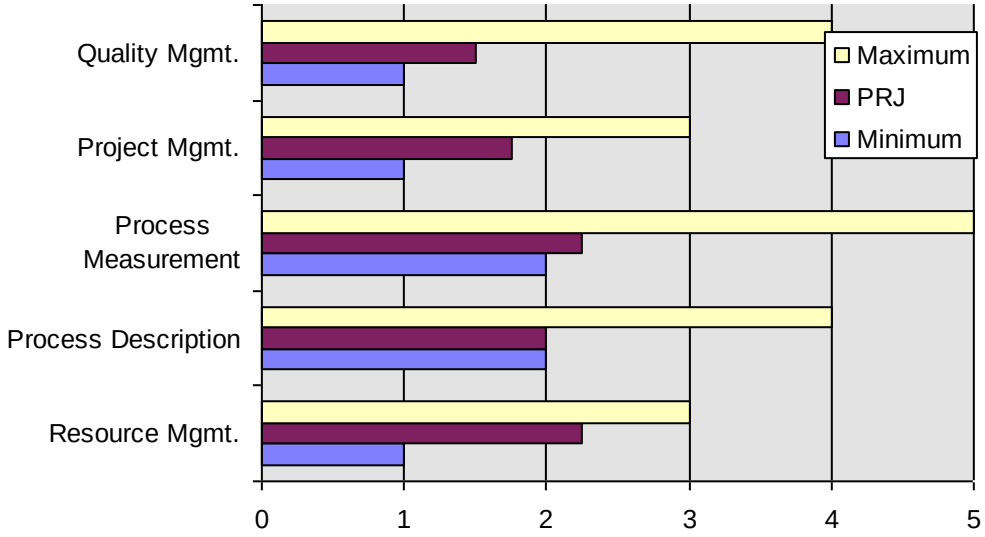


Figure 12: Picture with a Maturity Level Profile Including Min and Max Values

As mentioned in section 1 questions belonging to different maturity levels are assigned to one process attribute so that BOOTSTRAP processes span over many maturity levels. However, not every process attribute contains questions from all maturity levels 2 to 5. Resource Management, for instance, contains level 2 and 3 questions and thus can span from maturity level 1 to 3. Whereas Quality Management, for instance, contains level 3 to 4 questions and thus spans from maturity level 2 to 4. etc. Therefore in BOOTSTRAP version 2.3 it was always important to know the range (from minimum to maximum level) in which the calculated maturity level had to be interpreted.

Procedure:

Determine the lowest attribute in the profile : e.g. Quality Mgmt. = 1,5 $\Rightarrow$ START Level = TRUNCATE (1,5) = 1

For all levels from START to 4 do

sort the attributes between level START and START+1 by

d(actual level, maximum level).

The distance d(actual level, maximum level) denotes the improvement potential.

Applying the procedure to Figure 12 leads to:

1. Quality Management
2. Project Management

3. Process Description
4. Process Measurement
5. Resource Management

The priority is therefore defined by the highest improvement potential (where most of the activities are still missing).

Benchmarking Model

Another influence and model which is standard practice in process assessment consulting is to provide organisations with benchmarks and with mean profiles of organisations in the same industry sector.
These benchmarks are usually a decision support for top managers who want to first compare themselves with their competitors and with the market needs before starting to invest in any direction. The project managers and practitioners rather are interested in the technical installation of the improvement actions and implementation of new approaches, methodologies, and technologies.

This service is supported by the BOOTSTRAP database maintained by the BOOTSTRAP Institute.

Techniques for Supporting the Improvement Planning Process

After defining the priorities the following factors are still to be considered:

1. The improvement must be compliant with the company's business goals to ensure top management commitment and business benefit.
2. According to Pareto Analysis those 20% improvement actions must be identified which will bring 80% of the improvement benefit.
3. All projects must be measurement goal driven because without measurement you are not able to decide about success or failure objectively.
4. A proper improvement culture must be established which gives the people the feeling to actively participate in the establishment of a continuously improved organisation.
5. To ensure compliance with the BOOTSTRAP standard the standard template for action planning should be a framework for doing all this planning stuff.

Figure 13 illustrates the standard template for BOOTSTRAP action plans and shows which methodologies such as GQM (Goal Question Metric Approach), Ishikawa Diagrams [5] for factor analysis, ami [1] for establishing metricated goal trees to make improvement actions consistent with business goals, or priority definition techniques as presented in section 3 of this paper can be combined with this standard guideline.

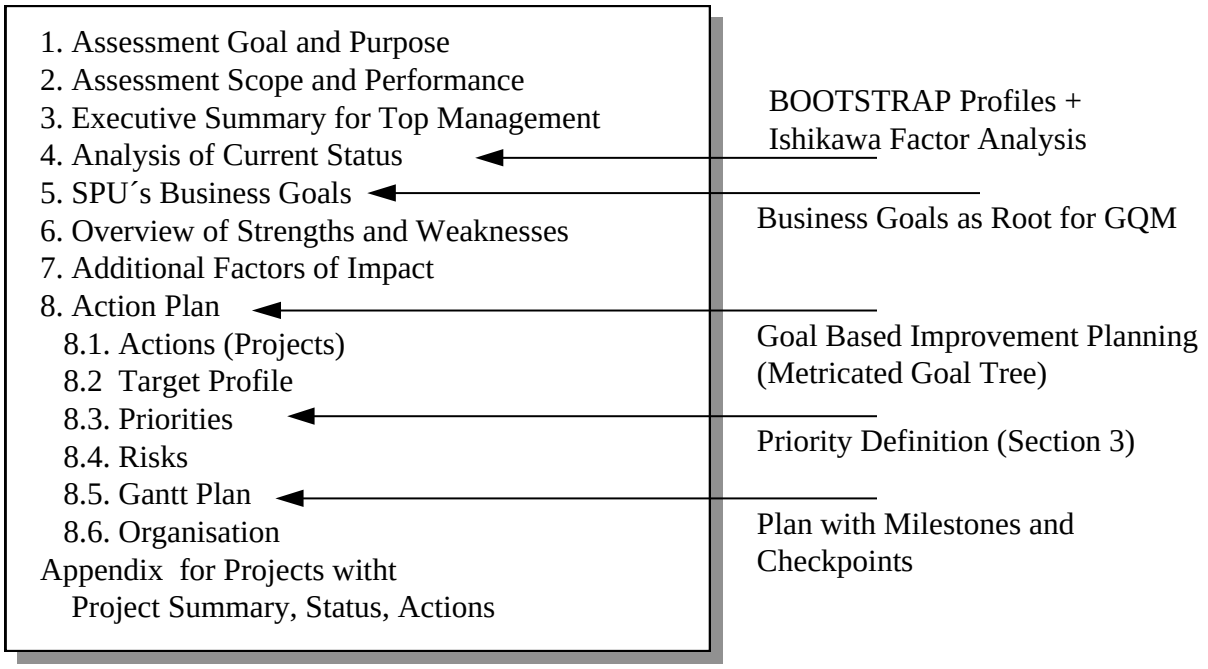


Figure 13: Picture with a Maturity Level Profile Including Min and Max Values

Conclusion

So far the developers of assessment methodologies prefer to present models, profiles, and process architectures. However, for organisations really implementing improvement the assessment effort is of course an important step but represents only 3-5% of the overall improvement effort. This paper wants to emphasise more the improvement than the assessment and points out that the proper definition of priorities is a key factor for improvement planning.

References

[1] Debou C, Fuchs N., Haux M.:ami: a Taylorable Framework for Software Process Improvement, In: *Proceedings of the second ISCN Seminar*, Vienna, September1995

[2] Dorling A., SPICE - Software Process Improvement and Capability dEtermination, Software Quality Journal, Volume 2, 1993, pp. 209-224.

[3] Haase V., Messnarz R., Koch G., Kugler H.J., Decrinis P., BOOTSTRAP: Fine Tuning Process Assessment, *IEEE Software*, pp. 25-35, July 1994

[4] Haase V., Messnarz R., Cachia R.M., Software Process Improvement by Measurement, in (ed.) Mittermeir R., *Shifting Paradigms in Software Engineering*, pp. 32 - 41, Springer Verlag, Wien, New York, Sept. 1992

[5] Haziza M., Minor E., Pofelski L., Blazy S., Software Maintenance - An Analysis of Industrial Needs and Constraints, in: ed. Marc Kellner, Proceedings of the Conference on Software Maintenance in 1992, pp. 18-26, IEEE Computer Society Press, Orlando, Florida, US, 1992

[6] Kuvaja, P., and Bicego, A., BOOTSTRAP: Europe's assessment method, IEEE Software, Vol. 10, Nr. 3 (May 1993), pp. 93-95.

[7] Kuvaja P., Messnarz R., BOOTSTRAP - A Modern Software Process Assessment and Improvement Methodology, in: Proceedings of the 5th European Conference on Software Quality, September 17-20, Dublin, Ireland, 1996

[8] Kuvaja P., Simila J., Krzanik L., Bicego A., Koch G., Saukkonen S., Software Process Assessment and Improvement, The BOOTSTRAP Approach, Blackwell Business, Oxford, UK, 1994

[9] Mehner T., Völker A., Siemens Process Improvement Approach, Proceedings of the ISCN '95 Conference, ISCN Ltd, Vienna, Austria

[10] Messnarz R., Kugler H.J., BOOTSTRAP and ISO 9001: From the Software Process to Software Quality, in: Proceedings of the APSEC´94 Conference, Computer Society Press of IEEE, Tokyo, Japan, 1994

[11] Messnarz R., Scherzer H., The Evolution of a Quantitative Analysis - the BOOTSTRAP - Approach, in : (eds. G. Chroust, P. Doucek) Proceedings of the Interdisciplinary Information Talks 95, pp. 198-213, Oldenbourg, Vienna, Muniche, 1995

[12] Paulk M.C., Curtis B., Chrissis M.B., *Capability Maturity Model for Software*, (ed.) Software Engineering Institute (USA), Technical Report CMU/SEI-91-TR-24, Carnegie Mellon University, Pittsburgh 1991

[13] Paulk, M.C., and Konrad, M.D., An overview of ISO´s SPICE project. American Programmer (February 1994).

[14] Paulk, M., et al. Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, Feb. 1993.

[15] Paulk M.C., Weber C.V., Garcia S.M., Chrissis M.B., Bush M., *Key Practices of the Capability Maturity Model*, Version 1.1, Technical Report CMU/SEI-93-TR-25, Software Engineering Institute, Pittsburgh 1993

[16] Rutschek G., FESTO - Experience with the ESSI Application Experiment ODB, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[17] Völker A., Software Process Assessments at Siemens as a Basis for Process, Improvement in Industry, Proceedings of the ISCN'94 Conference, ISCN Ltd., Dublin, Ireland

An 'IDEAL' approach to maturity increases with the SEI CMM

Éamonn McGuinness, *aimware Ltd., Ireland.* *eamonn@aimware.com*

Benchmark the Software Process Improvement Project

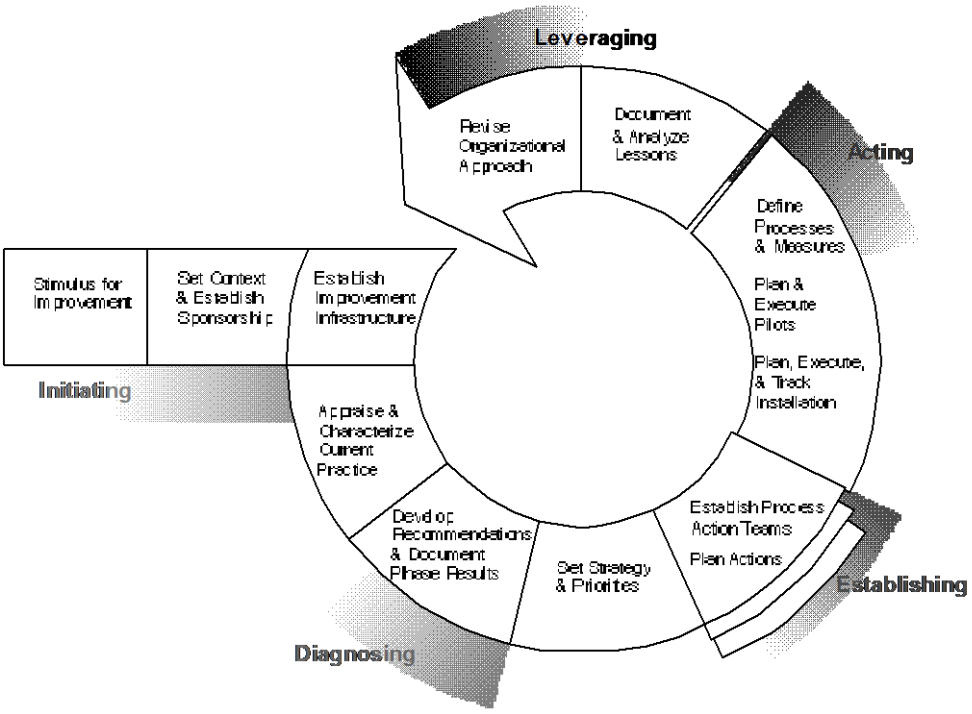
Is your wife faithful to you? Is your husband playing around?! You probably have seen the score sheets in magazines to help you answer the above questions! Well, here is the 'pop' check-sheet to help you figure out if your process improvement project is going to succeed. Alternatively you could read Watts Humphrey's famous book [1].

Process Improve *Factors Critical to Success*	*Your Score* *Yes or No!*
1. Strong, visible and active senior management sponsorship is needed	
2. A clear picture of the starting point is necessary (preferably against a recognised international model, e.g. SEI the CMM[sm] [2], [3](CMM is a Service Mark of Carnegie Mellon University), ISO 9001/TickIT)	
3. A focus on the results required is essential. Stretch goals do work and may be needed to rock some people from their comfort zones. A focus on the results needed should also keep the long term in mind (e.g. not just get ISO 9001 or SEI Level 2 by 1998 - i.e. invest for the long haul, for the capability to be gained).	
4. Make the improvement a project that is well managed. Since it is difficult enough to deliver on the day job (i.e. delivering software) it proves to be even more difficult to deliver on process improvement. The latter requires more careful project management or it will fail or deliver mediocre results.	
5. Ensure strong participation of as many of the people as possible. It is best if it is done by the people for the people! Imposed solutions tend to be resisted fiercely and may have a longer implementation schedule.	
6. Software Development needs an approach and so does Process Improvement. Select a lifecycle of phases and activities that will deliver on the desired result. A simple form of the Deming lifecycle works every time[4]!	
7. Commit real resources to the effort and if necessary re-prioritise some of the other work to get the improvement started. Engineers will really appreciate the seriousness of these actions!	
8. Take one bite of the elephant at a time. Don't allow too much to be tackled especially until some benefits are delivered. All of the models are broken into key areas, so there is no need to tackle them all at once.	
9. Provide automated support as early as possible. It is one way to help the institutionalisation.	
10. Have a vision of where you want to go after the initial improvements (whether onwards and upwards or stand still) and devise an appropriate strategy to meet this goal.	
Your Total Score out of 10 is →	

Ask yourself 10 questions based on the above points and the thought process will either confirm that you are on the right track or give you ideas on how to improve your process improvement efforts. If your score is low, then this paper may help with some practical hints and tips.

An 'IDEAL' Process Improvement Lifecycle

The SEI (Software Engineering Institute) is a US Government and Industry sponsored body who focus on products and technologies for software process improvement. They are located at Carnegie Mellon University. The SEI, who have contributed so much of their process improvement models, methods and results to the public domain, have recently released a process improvement model depicted here, called IDEAL[sm] [5] (IDEAL is a Service Mark of Carnegie Mellon University, US).



The IDEAL process improvement lifecycle description given in this paper is annotated with a case study of a proposed implementation in a real software company, aimware. The quality quest at aimware is interesting in that the company attempted to inject quality into the company at the start of the company lifecycle … which meant it was in place for the start of the product lifecycles. aimware is thus used here as a case study to explain the various stages of the proposed improvement lifecycle.

Case Study Background - aimware Ltd.

aimware are in the business of developing software for the software process. The company has developed a software product to help companies achieve higher levels of software process maturity based on the SEI CMM model. The product is being further developed in partnership with European-based US companies and indigenous companies. The aimware product suite consists of software which computerises the documentation templates and workflows of the software process. It also contains the option of a fully documented set of

processes which adheres to the SEI CMM and / or ISO 9001/TickIT models. Their customers include Telecom Eireann, CSK Software Ltd., Kindle Banking Systems Ltd. (part of the MISYS group), CBT Systems, Tellabs, EDS, Motorola, etc.. In summary then, aimware delivers software and processes in a box for the implementation of the software process.

Initiating Phase

In this sample 'IDEAL' Plan, the 'I'nitiating phase has three main steps:

Initiating
i. Recognise or get improvement impetus
ii. Set improvement business context & goals
iii. Ensure Senior Sponsorship is in place

Recognise or get improvement impetus

Companies typically need a jolt into software process improvement. This can come from increased customer pressure in the face of visible and mounting product quality problems. It can come from purchasers who mandate certain levels of quality against some international software quality model (e.g. ISO 9001 or SEI CMM Level 2 to bid for contracts). It sometimes comes from within, where companies recognise that they can not keep going the way that they are going! These companies have to get projects back in control … if indeed they ever were in control! In some companies it is a corporate dictate / objective that all subsidiaries gradually improve their processes.

The inference in all of the above is that the status quo is not good enough. The impetus is then the reason that companies enter into the improvement cycle. It is the pressure that starts companies on the road to improvement. It is very important to recognise this impetus and call it out loudly and clearly. It may not be the best reason in the world for starting process improvement but it is typically a real pressure that can keep the flame alive when other pressures mount and rise up against the process improvement project.

Set improvement business context & goals

Many improvements start, but few finish! Many of these failed initiatives start in the technical department as the technical staff know that they have to improve. However they sometimes forget to tie these efforts to the overall business goals. This means effort expended on the process improvement is the first project to be sacrificed in the face of other business pressures. Preventing defects reaching the customer is a good business goal to have! There are however many other goals that can tie software process improvement to the business in question. Most companies can examine their goals in respect of the Time, Quality and Cost variables. Indeed most companies while ambitious in all three directions are usually more focused on one of the three. For some companies (e.g. commercial software houses) time-to-market is critical and therefore reducing cycle-time is key. For other companies reliability (e.g. aircraft control software) is crucial and therefore minimum defect levels are the goal. Some companies are driven primarily by cost (e.g. in-house IT departments) so keeping projects to budget is critical. In most companies the driver is a combination of all three but it is important to understand where the real emphasis lays. In other companies the emphasis varies in the different software producing units (e.g. the Maintenance group focus on minimising defects whereas the New Products group might focus on time-to-market). It is essential to see what is driving the company and then ensure that the process improvement is "goaled" on delivering to or helping these overall objectives. If the process improvement can be seen to contribute to these overall goals it has a much better chance of prospering. Clearly, numeric, objective and measurable goals are best! Examples are provided in the next section.

Ensure Senior Sponsorship is in place

If the idea for process improvement did not come from the top of the organisation then it is essential to have a very senior manager sponsor the effort. If I might be so bold as to suggest, that one should seriously consider stopping the process improvement effort at this early stage if there is no sponsorship in place, or at least delay the project until the sponsorship is evident and visible.

What does a sponsor do?

- Resources the project (money, people, etc.)
- Gives the project direction, focus as well as visible and vocal support
- Reviews the project to ensure it is on target and delivering the results
- Changes the focus of the project as appropriate
- Ensures that the project has the support of the organisation
- Helps the project through difficult times (resource conflicts, etc.)

Case Study: aimware impetus, business context & business sponsorship

aimware impetus

aimware are in the business of process improvement - services but in particular, associated automation software. Having this expertise and knowledge meant they had
incentives, both internal and external, to implement their own Quality System and then have it certified. Internally because they knew the potential benefits of process improvement, and externally they felt that customers would recognise the value of working with a company who practice what they sell. In essence how could they justifiably sell the ideas, concepts, services and products of software process improvement if they did not “buy” them! They had to eat their own dogfood!

aimware business context and goals

Recent statistics from Watts Humphrey, the 'Edison' of the Software Process world, suggest that there are 100 defects, on average, produced for every 1000 lines of code written. Most do eventually get taken out during the software's life-cycle, some in compilation, some in testing, others during inspections etc. But some only make their way out, when they are least expected or wanted! The first key thing to notice about this defect rate is that it appears to be consistent across senior and junior engineers alike. The second is that there are differing costs associated with fixing bugs at the different life-cycle stages. Thus, a bug fixed at the design stage is much less costly than a defect encountered at the coding stage.

In addition to this, it is commonly recognised that testing is the least efficient means of getting rid of defects. Although testing is absolutely necessary in a properly constructed process, ideally it should serve only to catch the bugs not caught at earlier stages. Using it as a general method for catching and correcting bugs is costly and inefficient.

As a business manager it pays to have an effective means to deal with the problem of defects. Prevention is cheaper than detection and a good process allows the early capture of most defects. Thus, having a Quality System is more than just an added extra - it is crucial to the profitability of software development companies. That is why aimware ultimately are committed to having an improving software process. The current aimware objectives for software quality are included here by way of example.

The objectives are in six dimensions and their achievement or otherwise will be measured as described in the italics that appear in the parenthesis below:

I. ***Process***: Aim for and achieve SEI CMM Level 4 or higher by December 1999 *(as benchmarked on an official SEI assessment)*

II. ***Product***: Reduce post release defects by at least 50% from the baseline (Jan 1997) within 1 year, by at least 70% within two years and by 90% within three years *(to be verified by checking the software engineering database)*

III. ***Cycle Time***: Be capable of shipping 2 minor releases and 2 major release every year within 2 years of the baseline (Jan 1997) *(to be verified by examining the software engineering database)*

IV. ***Productivity***: Ship at least 30% more functionality in each major release than is shipped at the baseline date (January 1997*) (to be verified by examining the software engineering database)*

V. ***Cost and Benefit***: No major increases in headcount or technology spend over the current business plan to achieve the profit levels specified - (i.e. the above gains to be realised by increasing effort and focus on quality!) *(to be verified by examining the business plans)*

VI. ***People****:* Be capable of attracting and retaining the best people to aimware as the company will operate in a high quality working environment compared to the relative chaos of some software development environments. *(This is to be measured in terms of hiring costs and personnel turnover rates and associated costs by comparison to other local software companies)*

aimware business sponsorship

aimware are aware of the fact that process improvement is not cheap! They have invested heavily since formation in Quality. They have plans and budget to continue the investment. The key here is budget and plans, as having quality built-in as part of the business plan is essential. aimware recognise that process improvement has to be budgeted for and scheduled in the same way as any other successful element of the business. There is extra investment required up-front but it is believed that this is certain to lead to longer-term gain. "Short term pain for long term gain," says Fintan Manning, aimware Development Manager, in reference to this extra investment.

Summary - Initiating Stage

It may be possible to start a process improvement pilot without this level of sponsorship but serious company wide improvement is very unlikely to succeed without senior management sponsorship. Thus it is essential to ask the following three questions before exiting this stage … successfully:

i. Do we recognise or know where the improvement impetus is coming from?

ii. Have we set measurable improvement goals in line with the business direction?

iii. Is the requisite level of Senior Sponsorship in place?

Diagnosing Phase

In this sample 'IDEAL' Plan, the 'D'iagnosing phase has two main steps.

Diagnosing

i. Decide which 'measures' to take, e.g.

- *a. Process - e.g. CMM Assessment*
- *b. Product - e.g. Defects pre and post-ship*
- *c. Resource / Cost - e.g. Size and cost of projects*
- *e. Revenue - e.g. Cost and benefit*
- *f. Productivity - e.g. Size and / or cost over time*

ii. Take the 'measures'

Decide which 'measures' to take

It frequently happens that process improvement does indeed take place successfully in some companies. This usually happens when there is a push for ISO 9001 certification. Unfortunately the only measure taken is the binary measure of certification achieved or not! When this measure turns from 'No' to 'Yes', the impetus for process improvement sometimes fades. Similarly, although there may be a warm and fuzzy feeling about the certification, many companies can not quantifiably say what the improvement achieved in real terms. Some within the organisation say that it had been a waste of time and bureaucratic, while others say that it was a great benefit. And so the debates rage, fueled by emotion, charge and counter-charge … but no concrete evidence! With the latter in mind, it is best to measure the effect (or lack of effect) of the process improvement program, so that companies know whether or not there is a benefit to be gained from process improvement costs within their environs. Similarly it is difficult to chart an exact course for the process improvement project without knowing where the company is, with respect to process improvement. Unfortunately no one measure alone is sufficient and six types of measure are discussed here for companies to consider. With measures like these companies can find themselves on the process improvement roadmap and select a route to drive on.

Process Measures - e.g. CMM Assessment

A good process gives a company a very good chance (no guarantee!) of a resulting high quality product. There are now many international software quality models against which companies can benchmark themselves, e.g. SEI CMM, ISO 9001/TickIT, ISO 9000-3, BootStrap or SPICE [6]. The choice of model is maybe less important than the use of the model. Unfortunately many companies merely assess to get certified or to achieve a certain CMM level. While there are obvious business pressures to achieve certifictaion, when this is taken too far, the long term results are less than impressive. Staff tend to get disinterested as they see management more interested in the perception of a CMM level, than the reality of process improvement. Also one of the main purposes of the assessment (i.e. highlight areas that need improvement) is thwarted. A good assessment will deliver at least three results:

1. a benchmark against a recognised model (e.g. SEI CMM, ISO 9001, etc.)
2. a list of key software areas that need immediate improvement (12 months)
3. a group who have actively participated in the "assessment", reached consensus and who are motivated for the upcoming improvement.

I must confess a preference for assessing with the maturity based models, like the SEI CMM, where software groups are given interim targets to shoot for on the road to software engineering excellence.

Product - e.g. Defects pre and post-ship

As was explained above a good process implies a good chance of a good quality product, but no guarantee! To ensure that the software product is indeed of high enough quality, a software group should measure the number of defects found pre- and post-ship. These figures should start to improve as the process improvement program kicks in. A group will not know how much the product quality is improving unless it is measured from the start. These measures are also an indicator that the process improvement is working as planned and delivering the correct benefit.

Resource / Cost - e.g. Size and cost of projects

Of course it is possible to spend a lot of time and resource on process improvement and to have defect levels coming down … but at what cost? We could all probably get near zero defects if we had unlimited resources and time. More often than not we need to achieve improvements in process and product quality with the same resources. It is important therefore to measure the size, effort and cost of software projects. The size measure can be either the traditional "lines of code" or the newer "function points". Alternatively if you are convinced of neither and sick of the endless debates about the relative merits and demerits of each, then make up your own size measures! If you look long enough and hard enough you will find something! (see sample below) So what, if you are not using an industry standard measure and can not benchmark yourself against the "Top Ten" lines of code performers or the "Fortune 500" function point performers. All you really need is some way to compare your own performances (past, present and future).

Revenue - e.g. Cost and benefit

All this investment, time and effort had better make you more money, give you happier customers and keep your staff from getting frustrated with chaotic software practices. There is only one way to find out - measure!

Productivity - e.g. Size and / or cost over time

Productivity is the measure that helps you combine some of the above measures in an appropriate fashion. Some companies are interested in how much functionality (measured in terms of size) they delivered in this period of time compared to the previous period of time. Many companies investing in process improvement would like to see more software being delivered, with less defects, in a shorter space of time, with the same or less resources!

Take 'measures'

There are no right or wrong measures to take … but it's vital to get a balanced set that:

- accurately reflect past performance
- check to see if the goals set in the "Initiating" stage were too severe or too easy
- give a baseline with which to compare future performances against
- set the direction and pace for the required improvement

Getting a balanced set of measures implies measuring the different dimensions, e.g. process, product, cost, resource/effort, size, time, etc. This stage of the improvement (the Diagnosing) is the time to take these measures. Many companies start the improvement with no measures and of those that do measure, it appears that the majority measure in the process dimension only. Clearly the information may not be there for all dimensions but either a study can be made to extrapolate this information from the various sources within the company or a set of actions started to get this information measured over a period of months. Remember, we are not talking about measuring every thing that moves in the software process but we are talking about taking a few key indicators. And finally as you will doubtlessly have heard and read elsewhere, you must not use these measures to judge individual performance within your organisations, as you will soon be receiving measures that you can not rely on - and you would deserve no better!

Case Study: aimware measures

aimware started the Quality definition and improvement project as soon as the business plan was written for the company. Provision was included for time and money in the business plan to invest in Quality, the same way other companies invest in equipment and people (aimware invested in these also!). These initial endeavours were successful. As a result of these efforts, aimware are now registered to ISO 9001 / TickIT. The certificate is valid from May 7th 1996, which is not bad since the company was only incorporated on December 7th 1995. aimware have all the typical components of a software quality management system, that one would expect from an ISO 9001 / TickIT certificate, including but not limited to continuous improvement through audits and corrective actions. aimware sponsored an SEI CMM search conference in-house at the end of July 1996. This is a style of SEI assessment without a lot of the overhead of an official assessment. This gave aimware further ideas and consensus on what to improve and these improvements require five months to put in place (September 1996 to January 1997). These are essentially the elements necessary to go from ISO 9001 / TickIT to a strong SEI CMM Level 3.

At that stage (January 1997) aimware will be in a position to examine the records in the software engineering database, talk to the people on the ground and take the various measures, that accurately measure the past performance. This will enable aimware to re-calibrate the process, product, cycle-time, productivity, cost/benefit and people objectives set at the Initiating stage of the improvement lifecycle. A formal SEI CMM assessment will be the type of assessment carried out to measure the process maturity. This assessment team will also be tasked with taking the other 5 measures after the SEI CMM assessment, so that the results presented cover the other dimensions listed above also.

Summary - Diagnosing Stage

It is essential to have indicators like these before exiting this stage … successfully:

i. *Process Quality and Maturity*
ii. *Product Quality*
iii. *Cycle-Time (Time to Market)*
iv. *Productivity*
v. *Cost and Business / Profit Benefits*
vi. *People Factors*

Establishing Phase

In this sample ‘IDEAL’ Plan, the ‘E’stablishing stage has the following four steps:

Establishing
i. Set Strategy and Priorities (refer to CMM and business priorities)
ii. Finalise Improvement Infrastructure
iii. Establish Process Improvement Teams (PITs)
iv. Plan PIT team actions

Set Strategy and Priorities (refer to CMM and business priorities)

An organisation that set objectives as part of the Initiating stage of the process improvement lifecycle should now know if they were realistic or not. The process objective might have included the achievement of CMM Level 4 within twelve months, as they thought they were a strong CMM level 3. But when the process measure was taken in the Diagnosing stage, they might have turned out to be a rock solid CMM Level 1. This then is the time to re-calibrate the improvement objectives set at the Initiating stage against the real feedback of the Diagnosing stage. At the end of this step, the objectives or priorities have been finalised. Also at this step the improvement strategy will need to be set. How many key areas will be tackled? What key areas will be tackled first? What tool support will be used? What will the project structure of the improvement be? How will training be delivered?

Finalise Improvement Infrastructure

The technology infrastructure needs to be put in place. It has been the authors experience that long term institutionalisation of processes needs good tool support. People will adhere to best practices for the early part of the improvement but long term compliance requires automated support. This should not be so shocking to us software folk, who after all spend most of our waking working hours automating the business processes of others! Why should we be like the cobbler's children with holes in our shoes?! The technology infrastructure should include a software engineering database that can house process defining artefacts such as policies, processes, standards, procedures, templates, as well as process implementation artefacts such as plans, reports, defects, risks, issues, metrics, etc.

Establish Process Improvement Teams (PITs)

The improvements will not be delivered by accident. Resources need to be committed to the improvement. The improvement needs to be a real project with real people! It is best to staff these teams with people who want to improve and know something about the area in question. These teams need to be given some time and space to deliver on the required actions (e.g. on average a half a day each week).

Plan Process Improvement Team actions

If the improvements are planned there is a good chance (no guarantees) that they will be delivered! These people are likely to be very busy and committed to the 'real' work of developing software. Thus this improvement work will be playing second fiddle most of the time. This means that as soon as the pressure comes on for the 'real' work (maybe due to poor processes!) the secondary task of improvement will be shelved. It tends to be very hard for these people to switch back into the improvement team work after the latest crisis. Recognising this as fact, it becomes imperative to rigorously manage (i.e. plan and track) the improvement work as a mini project.

Case study: aimware priorities, technology infrastructure and resources

Priority
When aimware get to this stage they will re-calibrate the objectives as set out above in the Initiation stage against the results of the Diagnosis stage. From a process point of view, other sub-objectives to aim for may well be added at this stage as follows:

1. control project performance quantitatively
2. measure software product quality and set new targets for each project
3. manage the achievement of these measurable targets on each project
4. know the organisation process capability in quantitative terms
5. plan process management activities in quantitative terms
6. identify common causes of defects and prioritise them
7. plan and execute defect prevention activities systematically
8. achieve organisation wide involvement in continuous process improvement
9. systematically evaluate new technologies for quality & productivity improvements
10. incorporate the relevant new technologies into the organisation

Technology Infrastructure
aimware have a software engineering database capable of managing the software process assets (e.g. policies, lifecycles and procedures, etc.) as well as the artefacts created as a result of having these policies (e.g. defect reports, metrics, plans, etc.). This workflow enabled database not only helps engineers do a better quality job, it is the way the job is done, so process deployment and fidelity is ensured.

PIT Teams
Based on the aimware experience of making the large effort to get to ISO 9001 / TickIT, they needed one senior resource on the project one day a week and three other people 1 day every two weeks (or 10% of their time.) This serious commitment of resource is what gave aimware the early results. To achieve the next massive jump (e.g. SEI Level 4 or higher) aimware need an even more serious resource commitment for the project. It is estimated as follows:

-> 1 Senior Quality Project Manager	2 days / week for 3 years = 135 days
-> 10 Engineers, Marketing reps, etc.	0.5 day / week for 3 years = 675 days
-> 1 Senior Manager:	1 day / month for 3 years = 36 days

Allowing for just over 10% contingency, this is at least a 1000 person day project which is roughly equivalent to 5 person years over three elapsed years. Serious effort!

Summary - Establishing Stage

To successfully exit this stage you should have:

i. Set the Improvement Strategy and re-calibrated the Priorities
ii. Finalised the Improvement Technical Infrastructure
iii. Established Process Improvement Teams (PITs)
iv. Planned PIT team actions and committed real project resources

Acting Phase

In this sample ‘IDEAL’ Plan, the ‘A’cting stage has the following six steps for each key area to be improved:

Acting
i. Define process, tool support and measures
ii. Plan pilots

iii. Execute pilots
iv. Plan company/group wide implementation
v. Installation
vi. Support and Track installation

These six steps are likely to form the basis of the action plans or mini project plans for each of the Process Improvement Teams.

Define process, tool support and measures

Using their collective wisdom and some basic research (e.g. on the internet and through books and courses perhaps) the team will decide how best to perform the process in question. The process might be improved testing or improved project management or some such key area of the software process. The team will also be governed by the overall improvement objectives set earlier in the project. They will draft a new process or amend an existing one in order to describe how the process should be performed in future. The process will also illustrate how tools should support the implementation of the process and it will also explain how the process will be measured, to verify success or otherwise.

Plan pilots

The team will know that there are some rough edges in their new process and piloting the process on a real project will smooth these out. They also know that most projects are busy enough with the 'real' work, so that they need to plan in advance to get the processes piloted. This does not always pose a large problem if the team leaders are also on the Process Improvement Teams. They generally find the time to try out their new found ideas!

Execute pilots

The processes are tried on the designated projects and they usually run into some difficulty, so it is important to have a process coach ready to help out, so that the good is not thrown out with the bad, when the teams get frustrated.

Plan company/group wide implementation

What works in one area needs to be transferred to other areas - it will not seep out by accident! This step should go well if the different teams are feeding back to the larger group what worked and what did not work. At this stage it is a good idea to give the larger group the opportunity to review and improve the processes in question. They will see that the processes were useful and at the same time see that they had an opportunity to influence them. This helps them feel more ownership of the new processes.

Installation

The new processes are now ready to be rolled out to the other projects. Again it may not be possible to do this all at once, so a staged plan may be required.

Support and Track installation

One can not assume that the process will be in place overnight! Indeed it might not be used at all. It will be vital to support, coach and facilitate the new projects, when they are

installing the new processes. It will also be necessary to check (through audits, reviews, etc) how the new processes are being deployed. As a result of these checks you will find some combination of the following:

- people need coaching as they do not fully understand how to perform the process people need 'pushing' as they are not bothering to perform the new process
- the process needs changing to better suit the environment

Case study: aimware "acting" out process deployment

Any practices found useful or better than those aimware currently have, make it into the process database and all staff are trained and coached on their use, so that the practices become everyday processes. It should also be noted that aimware are all working on the one very large project with the sub-projects being frequent deliveries. This has the advantage that whatever is experimented is implemented and piloted by everyone straight away. It does not however mean that the practices are institutionalised immediately. It will be aimware's challenge to ensure that all practices are identified, evaluated and the good ones committed to the organisation process database and implemented for each of the subsequent projects and deliveries.

aimware have weekly meetings where Quality Improvement is a standing agenda item. This "slot" promotes the best practices of late and chases up on outstanding improvements. They have engineers on one sub-project auditing other sub-projects and better practices are replicated through these events. They have external audits with the ISO 9001 / TickIT registration and these audits also promote and disseminate best practices. They have a software engineering database where they store, review and refer to all company processes. Changes and improvements are managed and disseminated through this system. They have a constant focus on process improvement as was described earlier and this continues to disseminate the quality message and the best practices. After this project they will doubtlessly have even better internal dissemination practices as communication is a large focus of the higher levels of process maturity.

Summary - Acting Stage

At this stage companies need to tackle each required improvement in a systematic manner and the following steps seems as good as any the author has come across:

i. Define process, tool support and measures
ii. Plan pilots
iii. Execute pilots
iv. Plan company/group wide implementation
v. Installation
vi. Support and Track installation

Leveraging Phase

In this sample 'IDEAL' Plan, the 'L'everaging stage has the following three steps:

Leveraging
i. Analyse and Document lessons learned
ii. Consider taking a 'break'
iii. Start the next IDEAL Loop

Analyse and Document lessons learned

This phase can be as simple as it suggests, i.e. some form of post mortem to analyse the results achieved. The results should be pretty evident if the measures were in place as described in the previous sections. What will be less evident is how people feel about the exercise, what they saw as the advantages and disadvantages, the costs and the benefits, etc. It is very important to see what lessons can be learned or what should be avoided on the next turn of the process improvement wheel. It is also a good idea to publish any successes, so that credit can be given where it is due.

Consider taking a 'break'

It sometimes happens that organisations race up the software maturity ladder and get burned out very early on. It is often a better idea to let the improvements that were introduced really sink in to the organisation before driving for the next level. This is not the same as saying that an organisation can de-focus on software quality until the next great push. Quite the opposite, there will be a certain amount of effort and focus required from everyone to maintain the benefits realised during the recent push. It is a little like levels of fitness in that regard.

Start the next IDEAL Loop

Small improvements should keep happening at this stage, through whatever processes were put in place. But in order to get a real push for another level of process maturity and thus performance, it will be necessary to lead the organisation through another spin of the IDEAL wheel.

Case study: aimware will leverage the results

What lessons do aimware hope to have learned by this stage?
aimware want to show that it is possible to build Quality into the beginning of a company (and not just the beginning of a product / project lifecycle). They further want to show that dramatic results can be achieved if young (and maybe not so young!) companies adopt a Quality approach from the outset. They want to show that it is cheaper, easier and faster for companies to do this, than to wait until they are forced to do so by customers and poor product quality levels.

What are dramatic results? They were registered to ISO 9001 / TickIT five months to the day after the company was incorporated. They embraced Quality Day 1. It has been profitable for them to do so. They now want to AIM for higher levels of Quality and specifically SEI CMM Level 4?! Why?

- SEI CMM Levels 4 and higher are about measuring process and product and improving based on this quantitative feedback. aimware, like many companies, have good numerical controls on the business side - they now want the same on the Quality side of the business … to make more *profit*!
- It is important and relevant for their business.
- They are tired of people saying that Quality is impossible! As a group of professionals they have been helping companies improve their software quality for years and now they want to do it themselves … to show that it is not so impossible!
- To show that it can be done (and by a small company)
- Because it's out there!
- *It's profitable*: a professional way of executing and improving the performance & business
- *It's marketable*: a way of defining aimware apart from the competition

- *It's people conscious*: a mechanism to regulate the work rate, so that software staff have the time to produce first-class work and enjoy it (and not be always delivering to impossible schedules).

At the end of the project aimware will have the capability to:

I. Announce to customers that they are SEI CMM Level 4, thus giving them greater confidence in the companies capability and hopefully winning ***more business*** for aimware with them
II. Deliver a far higher quality product to customers thus ***reducing rework***
III. Be capable of shipping 2 minor release every year and 1 major releases every 6 months and thus be faster to market, thereby being ***more responsive*** to the rate of change of customers business
IV. Be far ***more productive*** by shipping at least 30% more functionality in each major release than is done today, thus delivering value for money to customers
V. Incur no major increases in headcount or technology spend over the current business plan to achieve the profit levels specified - (i.e. the gains to be realised by increasing effort and focus on quality!) which will mean ***remaining competitive***
VI. Attract and retain the best ***people*** to the company, as this is the key to achieving all of the other commercial objectives!

What next? Assuming aimware do reach SEI Level 4, they will be in a natural optimising or improving loop. The actual improvement decisions that will be taken, will be decided based on the data that is being returned from the process and product quality. They will have a process defined for continuous process improvement company wide. Most improvements should at that stage come from the engineers through this process. Assuming that this all works according to plan, aimware will take a break and let the improvements really sink in. They may well look at ISO-SPICE after a few months or a year, if the business justification is there. Another model for continuous improvement that particularly interests them is the People Capability Maturity Model. All in the future!

Summary - Leveraging Stage

The following simple steps form the core of the Leveraging stage.

i. Analyse and Document lessons learned
ii. Consider taking a 'break'
iii. Start the next IDEAL Loop

Results Achieved - The proof is in the eating!

aimware used the lessons above as the basis for the initial process improvement spin of the IDEAL wheel (December 1995 to December 1996). By using these lessons aimware were recommended for ISO 9001 / TickIT certification on their 5th birthday (5th month of operation not 5th year - and not 5 months from the start of the ISO project - 5 months from the incorporation of the company!!).

We will have to wait a few years to see exactly how aimware fare out when using this approach the second time out, as described above!

Summary and Conclusions

Many process improvements have failed - way more than succeeded. Looking at the success indicators at the start of the paper, it is possible to identify the ingredients of a profitable process improvement approach. The sample IDEAL plan combines these and other required attributes into a lifecycle of stages and steps to follow to successful and profitable process improvement.

References

[1] Humphrey, Watts S.: "Managing the Software Process", Addison-Wesley, ISBN: 0-20118095-2

[2] CMM: Paulk et al, "Capability Maturity Model for Software, Version 1.1", Report CMU/SEI-93-TR-24, Software Engineering Institute, Pittsburg

[3] CBA: "CMM Based Appraisal, SEI Lead Assessor Training", Course Materials, Software Engineering Institute, Pittsburg

[4] Deming, W. Edwards 1986. "Out of Crisis", MIT Center for Advanced Engineering Studies, Cambridge, MA.

[5] Peterson, Bill: "News, Software Engineering Institute", Software Process Improvement & Practice, August 1995, Pages 68-70, ISSN: 1077-4866

[6] Rout, Terence P.: "SPICE: A Framework for Software Process Assessment, Software Process Improvement & Practice", August 1995, Pages 56-66, ISSN: 1077-4866

The Level 4 Software Process from the Assessor's Viewpoint

Ken Dymond
Process Inc US
P.O. Box 1988
Annapolis, Maryland USA 21404
Internet: Processinc@aol.com

ABSTRACT
for an invited paper at the December, 1996 Conference of the
International Software Consulting Network
Brighton, U.K.

From the time the Capability Maturity Model (CMM) was developed (late '80s, early '90s), it was a matter of speculation whether software organizations existed or would be assessed at the higher maturity levels, levels 4 and 5. In the last two years, a few organizations with processes at these high levels have been reported. But most assessors (including compilers of the CMM) have never seen a Level 4 or Level 5 company. This paper describes a Level 4 company (one therefore in the top 2% of all organizations ever assessed) from the viewpoint of an outside assessor. Features that make this particular Level 4 process a paradigm case of implementing the CMM are described as well as innovations that go beyond the CMM and contribute to noteworthy robustness of implementation (institutionalization). The description serves as a brief case study of Level 4 practices and as an example for other organizations following the CMM road map. One hopes that with time a whole series of case studies will appear as organizations climb the maturity scale to Levels 4 and 5 and choose to make known how they implemented the CMM in their business environments.

Section 1. Introduction

The SEI's Capability Maturity Model for Software has 5 maturity levels.[19] Since the early days of the CMM's development and its use for assessing software processes, it was a hypothesis that organizations would be found at each level.[20] In 1989, SEI's first published data on the maturity profile of assessed organizations showed few above Level 2

[19] Mark C. Paulk et al. *Capability Maturity Model for Software, Version 1.1* (CMU/SEI-93-TR-24). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February, 1993.

[20] Watts Humphrey, personal communication.

and none at Levels 4 or 5.[21] Assessment results accumulated through 1991 still showed no organizations above Level 3.[22] Recent SEI data show a few companies at Levels 4 and 5, about 2% of all assessments reported.[23]

Evidently these high maturity organizations are a rare species. Few assessors and professionals in software processes have ever seen a Level 4 or 5 organization. (If you happen to be an assessor, your experience is likely to be mostly with Level 1 processes.) Since there were no Level 4 organizations assessed at the time the CMM was written, its compilers had to envision what such an organization would be like. It is instructive to compare a living example with the imagined construction.

The Level 4 software process should exhibit routine and effective use of all the key process areas of Levels 2 and 3 plus a pervasive understanding and control of processes and products by means of quantitative data. Thus, well-regulated processes under control of metrics should be the picture obtained by an assessment team in a Level 4 software-producing company.

The paper is organized as follows. Section 2 sketches the business environment of the Level 4 organization called X Company and describes how the company has implemented certain Key Process Areas (KPAs).[24] To stay within the compass of a short paper, I have highlighted only certain KPAs, those which displayed innovative features that seemed to go beyond the general recommendations in the CMM practices. Aside from the space constraints of a short paper, the requirements of non-disclosure of proprietary information limit more detailed descriptions of processes and tools. Section 3 summarizes observations on the rare life form of a Level 4 company.

The assessment method used to appraise the software process at the two sites involved was the CMM-Based Appraisal for Internal Process Improvement (CBA-IPI)[25] as defined by the Software Engineering Institute, Pittsburgh, Pennsylvania, USA. CMM version 1.1 was the process benchmark.

Section 2. Processes in a Level 4 Company

In this section a short sketch is given of the business environment and process improvement history of the case study organization, called in this paper, X Company. This

---

[21] Watts Humphrey, David Kitson and Tim Kasse. *The State of Software Engineering Practice: A Preliminary Report* (CMU/SEI-89-TR-1). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February, 1989.

[22] David H. Kitson and Steve Masters. *An Analysis of SEI Software Process Assessment Results: 1987-1991* (CMU/SEI-92-TR-24). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, July, 1992.

[23] Software Engineering Measurement and Analysis Team. *Process Maturity Profile of the Software Community 1995 Update*. Copy of presentation transparencies. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, November, 1995.

[24] KPAs and other elements of the CMM are described in Mark C. Paulk et al. *Key Practices of the Capability Maturity Model, Version 1.1* (CMU/SEI-93-TR-25). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February, 1993.

[25] CMM-Based Appraisal Project. *CBA-IPI Lead Assessor's Guide* v1.0. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, May, 1995.

is followed by descriptions of process areas implemented in a particularly striking way, especially processes interlinked for mutual support and synergy.

Section 2.1. Business Environment and Evolution of Process Improvement at X Company

X Company is located at two sites in India 1000 km. apart with a total professional staff of about 280. The company produces commercial software products for a worldwide financial market of banks and insurance companies. These commercial products operate on a variety of platforms. The company began as a spin-off in the early 90s of an Indian software company that is an internal division of a large international bank with headquarters in America. The founders wanted to sell products to customers in any financial market, not just to internal customers of the international bank. To do so, they had to incorporate separately from the international bank. X Company is majority-owned by Indians, though it retains the logo of the American headquarters company which is part owner.

Process improvement has been a part of X Company's business strategy since its founding. For example, the spin-off company adopted the standards and procedures manual of the parent company. Also, the founders of X Company, all software-knowledgeable, ensured that as the company grew, new hires would understand and follow the standards and procedures manual. The company's Chief Executive Officer (CEO) and founders determined that continuous quality improvement would characterize X Company's growth. By 1994, after examining ISO 9000 and the CMM as possible improvement scenarios, the company decided to follow the CMM as a road map for long-range process improvement. A Software Engineering Process Group (SEPG) function[26] existed from the early days. The motivation for an assessment was to have an independent check on what management and employees felt was a beneficial program. In other words, the improvement program was normal business practice, producing a return on investment (ROI) of more than 2 to 1; the assessment was not imposed by external customers but was to be an internal verification of process maturity level.

Section 2.2. Processes in a Level 4 Company as Seen by the Assessment Method.

This section gives a snapshot of certain key process areas and use of metrics as seen in this case study organization.

Key Process Area: Training

At the beginning of each year, X Company's upper management and the Training Coordinator estimate the organization-wide training needs and outline a training calendar of in-house and external courses to satisfy them. Special skills needed for upcoming projects, the annual business plan, and individual training needs are inputs to constructing the calendar. Individual training needs come from each employee's Individual Learning Plan or ILP. The ILP is a career-spanning document accumulating over time and filled out by both the employee and the employee's supervisor at the time. It lists the person's training needs as his or her career unfolds and records that the training was actually received when needed. The ILPs, as one source of input to the Training Calendar, are audited by SQA to ensure training delivery. The Training Calendar is then used to plan a training budget.

[26] The SEPG function in a software organization coordinates process improvement. See below under "Key Process Areas: Software Quality Assurance (SQA) and Organization Process Focus (SEPG)."

Most training courses are developed and given in-house. People who become proficient in a function like project management give training on that topic and develop or enhance the training course. Users or adapters of new processes or technology become trainers in that area. Everyone at some time thus has experience as trainer and as trainee. New hires receive mandatory induction training specialized differently for young people taking their first job fresh from school or for experienced people hired from another company. This induction training, which may include banking and financial applications if these are unfamiliar to the new hire, takes 6 to 8 weeks.

Planning and delivery of training for both sites is accomplished with a full time training staff of just one person, the Training Coordinator. There is no specialized cadre with full-time training responsibilities. Of special interest to an outsider, the training process was organized as a self-sustaining system for spreading expertise and lessons learned within the company. The question of outsourcing such a training function does not even arise in a case where the function is spread over all employees in the company and serves as a tool for reusing knowledge and for technology transfer.

People accept their roles as trainer or trainee as a normal job task, and the uniform coordination of all employee training is a major contributor to the sharing of processes and culture between the two sites. The resulting similarity in software and management practices between sites was easily noted in the assessment.

Key Process Area: Organization Process Definition (OPD)

The OPD key process area is abstract and one of the hardest to visualize among the 18 process areas of the CMM. A company fulfilling this KPA uses a standard set of processes so that each software project will organize itself in a repeatable fashion to reuse best practices the company considers vital. At the same time each project should be allowed to customize within agreed limits how it carries out the standard company approach for its own requirements of customer, platform, schedule, etc. The idea is to have both a standard way of operating for all projects, as a stable basis for improvement of the whole organization, and to have a systematic way of customizing the standard for project peculiarities.

X Company's solution to having a general standard and also a standard way of deviating from it is to define a set of core processes that all projects share and to define a number of different project life cycles in use. As part of a project initiation process (see below under Project Planning), the appropriate life cycle is chosen. Core processes include planning, construction (development or maintenance), configuration management, etc. Pre-defined project life cycles include rapid application development, client-server, etc.

Employees use the company's standard processes in the form of templates, process descriptions, procedures, and "how-to" guides in an on-line environment. The standard process is available through the same interface on everyone's desk top, so that processes are largely automated. Automation makes it easier to follow the standard way of doing things by starting from a fill-in-the-blanks template with on-line help and from examples of documents from previous projects. The automated interface to the process is also the employee's interface of record to the rest of the organization, that is, to one's colleagues and managers. The on-line process library, called QuBase for Quality Base, is also a source of training with tutorials and filled-out examples.

Though QuBase provides a standard way of discussing, deciding and recording work issues, paper documents do not disappear. The automated process interface did not prevent people from communicating face-to-face or by telephone or in writing. The templates and final versions of implementation documents were on-line; the in-process documents were on

paper and recorded the details of a project's life cycle -- audit comments, signature approvals, test results, meeting minutes, etc.

Key Process Areas: Project Planning, Intergroup Coordination, and Integrated Software Management

These three CMM KPAs are closely related in a Level 3 or higher organization and so are discussed together. X Company has a Project Office, staffed by a senior manager who is also the quality director, but the function itself is carried out by a number of managers. The Project Office is a focal point that oversees projects from start to finish.

A project begins by going through an initiation process. The initiation templates in QuBase require estimating and recording information on the technical aspects of a potential project (e.g., software size in function points and task complexity, functional requirements or methods of requirements elicitation) as well as on non-technical aspects (e.g., needs for: training, hardware and software platforms, office space, special equipment, travel, types of skills and number of workers, etc.). QuBase converts these non-technical requirements of the potential project to e-mail messages which are sent to the appropriate managers (directors of Finance, Facilities Management, Administration, Training Coordinator, Process and Quality Management -- Software Engineering Process Group or SEPG and Software Quality Assurance or SQA). The customer requirements and acceptance criteria for the project's deliverables are also recorded. For a new standard product (as opposed to a custom software product for a specific client), the International Marketing group acts as the customer and defines the acceptance criteria. Only after project risks have been assessed and all the managers agree to their commitments is the project authorized to consume resources, signaled by the assignment of a project identifier.

X Company has built a smoothly operating commitment process[27] for coordination among groups who support many projects with office space, computers, skills, travel, and technical functions like reviews and testing. Tracking of fulfillment of project commitments and the status of project risks as well as progress on overall business goals is done in company Quarterly Reviews of annual goals, monthly Senior Management Reviews across projects, and scheduled project status reviews. One of the topics in all these reviews is progress on the quantitative quality goals of the organization. (Goals for 1995 included reduction in defects by 50% and increase in productivity by 16% in all projects and functions.) Project performance is managed and measured against quantitative parameters (threshold or control limits on software size, effort, and schedule slippage). Risk management work sheets describing risks identified at project initiation as well as subsequent reviews and mitigation plans to offset risks are monitored. SQA plays a part in all reviews, raising issues that must be escalated for upper management attention. The results of review meetings, including issues raised and their disposition, are posted in QuBase and available to everyone in the company.

At a project's end, project closure templates describing lessons learned on the project and recommendations for improvements to company standard practice are filled out by the project leader and recorded in QuBase and thus made available to company staff. Process innovations, technical and management lessons learned and project quantitative data are thus accumulated in what the CMM would call a process library and process database.

---

[27] For a description of the commitment process see: Watts Humphrey. *Managing the Software Process*. Reading, Mass.: Addison-Wesley, 1989, pp. 69-82; and Kenneth Dymond. *A Guide to the CMM*. Annapolis, Md.: Process Inc US, 1995, pp. 2-31 through 2-36.

The result of practicing such a smooth and automated commitment process is that project problems are foreseen early and there are few crises. Though the messages generated automatically to company groups at project initiation time were process elements noteworthy to outsiders on the assessment team, X Company people pointed to the training in team building and group dynamics required of everyone as a major factor in their culture of cooperation.

Key Process Areas: Software Quality Assurance (SQA) and Organization Process Focus (SEPG)

Quality assurance, the job of SQA, and serving as the focal point for company processes, the job of the SEPG, are separate functions in the CMM, but the relation between them is often problematic in implementation. X Company structured the roles of these two groups to be mutually supporting. The SQA group reports independently of projects to the Process and Quality Manager. The SEPG is similarly independent of projects and reports to the same manager, who also heads the Project Office.

SQA conducts process audits at project milestones and reports results to the Project Leader and at the monthly Senior Management Review meeting. SQA audits all project artifacts, starting at the project initiation phase when the project plan is being formed. SQA also audits product testing and must certify a product as ready in order for it to be released. The release certificate depends on a numerical rating computed as a function of measures of test coverage, errors discovered, errors fixed, and requirements satisfied. SQA also certifies entry and exit criteria for the project to proceed to construction (development) and to test phases based on data specified in the quality plan section of the project plan. SQA is a member of the Change Control Board which, among other configuration management functions, determines the level of regression testing required for product modifications.

SQA also audits the SEPG's activities. The unanswered question of the CMM "Who shall audit SQA?" (like the ancient Roman question, "Who shall guard the guards of the state?") receives the answer in X Company: the SEPG. SQA audits every other function (including the Financial Dept. and International Marketing) and the SEPG audits SQA. X Company people refer to their "culture of compliance," natural to the banking environment, as one factor in their comfort at being audited. Another factor, noteworthy to outsiders, is the staffing of SQA by rotation from projects and other functions. There are no career positions in SQA. People serve for a time as auditors of their colleagues and then in rotation assume project or functional roles that are audited routinely by their peers.

The SEPG has a complement of 5 full-time people assigned out of approximately 280 total over 2 sites. Its roles include process awareness (induction training on X Company's processes), internal assessments, following and enhancing its own internal SEPG process (audited by SQA), and assisting projects to use pilot processes and technology. Like SQA roles, SEPG positions are not permanent but rotate among employees.

According to the CMM, process improvement should include participation by everyone, not just by the people assigned full time to the SEPG. Aside from participating in pilot process changes in a project or function, people become part of the process improvement effort by contributing technical lessons learned called "ECNs" (for Environment Capability Notes) to the on-line QuBase. The typical ECN is written by someone who has discovered the solution to a problem encountered in a commercial software tool (editor, compiler, etc.) used by the company. This was an example of reuse and of lessons learned.

Metrics

Though not a separate KPA, metrics are a theme in all KPAs (through the "Measuring and Analysis" practice in every KPA), and use of metrics data is expected to be mature (routine, effective, and efficient) in a Level 4 organization. X Company exemplified this maturity by collecting a handful of standard metrics on all projects, and then making the results known for projects and the company as a whole. Product defect density (from peer reviews and tests), effort data (collected via an internally developed tool, "PROMOTR", described below), software size (in terms of function points, lines of code, and module complexity), and schedule elapsed time and slippage are measured and recorded routinely. The data is summarized and analyzed in standard reports that everyone uses. Therefore the whole company had the same up-to-date knowledge of trends in defect density, defect origin by project phase, or productivity. Everyone viewed processes and spoke about them in terms of the same metrics, from the CEO to the most recent new hire.

One of the SEPG's crucial responsibilities is maintaining the company processes in QuBase and making process data available. The latter task involves collecting metrics data on projects and activities, which means making it easy for people who generate data to record it. Automation enables collection of effort data (time spent on a task) through the "PROMOTR" tool which interfaces to a widely used commercial project planning tool. The project leader develops and maintains in the planning tool the tasks, responsibilities, and schedule for an entire project. PROMOTR is networked to the desk top computer of everyone working on the project. A worker enters the number of hours spent on a task into PROMOTR, which uses the project identifier to insert the data into the relevant portion of the on-line project plan. As tasks are completed (for example by passing a planned review audited by SQA), PROMOTR causes the updating of the project plan for milestones achieved.

Project leaders complete a monthly Metrics Action Plan (MAP) report to track quality objectives (both project-specific and organization-wide) on their project -- for example, data on actual and projected defect and productivity rates. The SEPG compiles MAP reports for the whole company (both sites) and publishes a monthly QPCA (Quantitative Process Control Analysis) report within the company. This report tracks progress on X Company's quantitative goals for process improvement and product quality. Far from fearing the use of metrics data, everyone the team interviewed at the two sites said how much they depended on the MAP and QPCA reports to see how well the company's processes and their particular activity or project were faring. If a project's results to date fall outside control limits in the QPCA report, the project will identify the cause and describe actions it will take to correct the situation.

Finally, the release of a product was not determined by reaching a schedule date or passing the test phase or achieving sign off by department heads. Product release was the achieving of a product rating as certified by SQA encompassing all those things, but primarily the quantitative thresholds for defect density, process quality (all review issues resolved), requirements coverage, documentation, etc.

Section 3. Summary and Conclusion

The CMM, though quite detailed for a process standard, describes recommended practices in general terms. The CMM gives little information on Level 4 (2 KPAs out of a total of 18 for all Levels). It is hard to visualize the look and feel of "Level 4ness", historically rare, so that few people, including the authors of the CMM, have seen it.

It is noteworthy that a Level 4 company when seen through the formal instrument of the SEI assessment fulfills what the CMM predicts: Level 2 and 3 KPAs solidly in place and a marked understanding of software processes in quantitative terms. More striking is that the

Level 4 company implements those KPAs in unexpected ways that go beyond the CMM. A few of the innovations beyond the CMM have been highlighted above:

1) The training function for more than 200 people delivered with a full-time staff of one and little outsourcing because most training courses are internal, with the instructor roles distributed over the entire staff.

2) The rotation of people from projects through the SQA and SEPG so that these latter roles are not proprietary but filled by peers.

3) The smooth functioning of a commitment process coordinated by a project office and supported by an automated process environment (QuBase).

4) The uniform use of the standard process even across sites a thousand kilometers apart. This is due in part to having the process environment on line at one's desk top (easier to follow the standard process than to invent one's own) and networked to the rest of the organization. It is also due to the evident care to have the two sites feel equal (senior managers and staff functions like SQA, SEPG, training, senior management are distributed over both sites). A further cause of uniformity is that the standard process was not imposed but has been evolved by X Company people themselves over time.

Underlying the achievement of these notable features of the Level 4 company was something less tangible: the commitment of senior management to quality by means of continually improving processes. Process improvement is seen as a method for managing profits and ensuring jobs and therefore as an element, and a critical one, of business strategy. The Level 4 processes, no matter how impressive to an outsider, are not just fulfilling the CMM but make a difference in the business. The CEO expressed it this way: "We invest the equivalent of 13 people in process improvement for a year and we get back the productivity of 26."

References

CMM-Based Appraisal Project. *CBA-IPI Lead Assessor's Guide* v1.0. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, May, 1995.

Kenneth Dymond. *A Guide to the CMM*. Annapolis, Md.: Process Inc US, 1995.

Watts Humphrey, David Kitson and Tim Kasse. *The State of Software Engineering Practice: A Preliminary Report* (CMU/SEI-89-TR-1). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February, 1989

Watts Humphrey. *Managing the Software Process*. Reading, Mass.: Addison-Wesley, 1989, pp. 69-82

Watts Humphrey, personal communication.

David H. Kitson and Steve Masters. *An Analysis of SEI Software Process Assessment Results: 1987-1991* (CMU/SEI-92-TR-24). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, July, 1992.

Mark C. Paulk et al. *Capability Maturity Model for Software, Version 1.1* (CMU/SEI-93-TR-24). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February, 1993.

Mark C. Paulk et al. *Key Practices of the Capability Maturity Model, Version 1.1* (CMU/SEI-93-TR-25). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, February, 1993.

Software Engineering Measurement and Analysis Team. *Process Maturity Profile of the Software Community 1995 Update*. Copy of presentation transparencies. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, November, 1995.

Acknowledgments

I would like to thank the following people for their support of this paper. Any errors are the responsibility of the author.

Cynthia Wise, vice-president of Process Inc US and lead assessor on the site #1 CBA-IPI at Bombay, contributed not only her leadership skills to the assessments but also her ideas and experience to this paper.

Mr. Vivek G. Govilkar, Process and Quality Manager at Citicorp Information Technology Industries, Ltd., Bombay, kindly gave permission to describe some of the remarkable processes seen by the assessment team.

The anonymous reviewers of ISCN 96 gave their very helpful comments and suggestions.

Trillium - Using a CMM Based Benchmark for Software Product Procurement in the Telecom Sector

François Coallier

Bell Canada - IT Procurement,
2665 Roland Therrien, Longueuil,
Québec, Canada J4N 1C5,
fcoallie@qc.bell.ca

Abstract.

The procurement of software products entails risks that need to be assessed and managed. Software products are essentially complex and usually have (and are expected) to evolve during their life-cycle to follow changes in business needs. Also, most software products are bought either with some customization or before their development is completed. This paper presents the approach developed and used by Bell Canada to address this issue.

Introduction.

Bell Canada, a major provider of telecommunication services, is purchasing for about 1.3 billion Canadian dollars of software based products per year. These include the outsourcing of all of its Management Information Systems (MIS) development and maintenance as well as computer based telecommunication products.

To assess and manage the risks associated with the procurement of these type of products, technical expertise was brought in and developed within Bell Canada's Purchasing organization.

Initially a Software Quality Assurance (SQA) function, this technical arm of the Purchasing organization evolved into a "Quality Engineering" group. This group was then merged with proper purchasing expertise to become a specialized "Information Technology (IT) Procurement organization.

In the last 15 years, Bell Canada has developed processes tools and expertise to address the procurement of software products ranging from complex telecommunication products to management information systems (MIS). Our focus has moved from the classical "quality audits" that were done as part of vendor surveys to context specific vendor capability and product quality assessments focused on procurement needs.

While the traditional supplier quality management function was downsized and integrated in the Purchasing organization, a small group of specialists was put together to address specifically the management of software products' procurement risks.

Concurrently, a risk assessment and management process embedded in its procurement practices were developed and implemented. This process started with the issuance of Request for Quotations and is ingrained in all of the purchasing process, including the management of supplier portfolios.

Specificity of Software Products Procurement.

The procurement of software products requires a specific risks management process. This is due to:

- The high complexity usually associated with these type of products.

- The fact that, in many cases, the product that is purchased is still in development.

- Software products are usually expected to evolve as the business and operational environment of the product change.

High complexity is inherent to the nature of software products. This means that such products are not easy to develop and maintain. Many software products include custom designed hardware.

Often software products are bought while still in development, and/or that some customization is needed. In some case, the product could be mostly custom made (the exception to all of this is shrink-wrap software). This implies that risks associated with the delivery and the quality of the final products can be incurred. This is compounded by the complexity of the product.

Software products need to evolve in response to changes in business environment or practices, or changes in technology. If the product becomes difficult to enhance, significant costs can be incurred. At a certain point, the product may need to be partially or completely re-designed, or replaced by another one. This attribute of a software product is referred to as "maintainability".

All the above implies not only many potential risk elements, but also that the procurement of such product is associated with the establishment of a relationship with the supplying organization.

The procurement of these type of products must include a Due Diligence process that covers adequately the risks associated with these characteristics of software products.

Acquisition Process.

Our acquisition process [Fig. 1] includes all the activities from the product requirements up to contract management and operation. The following paragraphs describe, in generic terms, the sequence of activities involved during our acquisition process. This acquisition process contains eight steps and is in line with common practices as described in the ISO/IEC-12207 standard.

Need - The first activity is the expression by an internal group of a need for new features or services.

Requirement - The second activity is the documentation of the requirements for the software product to be acquired. This is done by a subject matter expert (or design authority) on behalf of the internal group.

Quotation - Using that requirement document accompanied by general company requirements, a Request For Quotation (RFQ) is issued to the vendor community.

Pre-selection - Using the answer from the vendors, the acquisition agent and the subject matter expert makes a pre-selection based on compliance to requirements and all other business issues. This pre-selection usually narrows down the list to a few suppliers.

Due Diligence/ Risk Assessment - Risks associated with dealing with the pre-selected vendors are then assessed. These include commercial, functional, operational and technological risks. Technological risks are assessed through Capability, Product and/or Project assessments as required. Site visits are usually done for only one vendor.

Supplier selection - The selection of the supplier is a business decision that requires gathering of all pertinent information. All of the relevant issues must be considered . Failure in doing so may results in a less than optimal decision, that could be proven costly during the life of the product.

Contractual agreement - The agreement with the supplier details, the product functionality, but can also cover product improvement goals and process improvement goals. Including them in the contract commitment towards product and process improvement insure the implementation of such programs.

Contract management and operation - This covers essentially the management of the relationship with the supplier. It includes support of internal users, resolution of product and service quality issues, contract updates, monitoring of process and product improvement commitments, etc... Follow-up assessment can be conducted if required.

The scope and effort required by each of these activities depend on the criticality and the nature of the product being acquired. For each new acquisition a customized process derived from our generic acquisition process is defined. This insures that resources are allocated where needed.

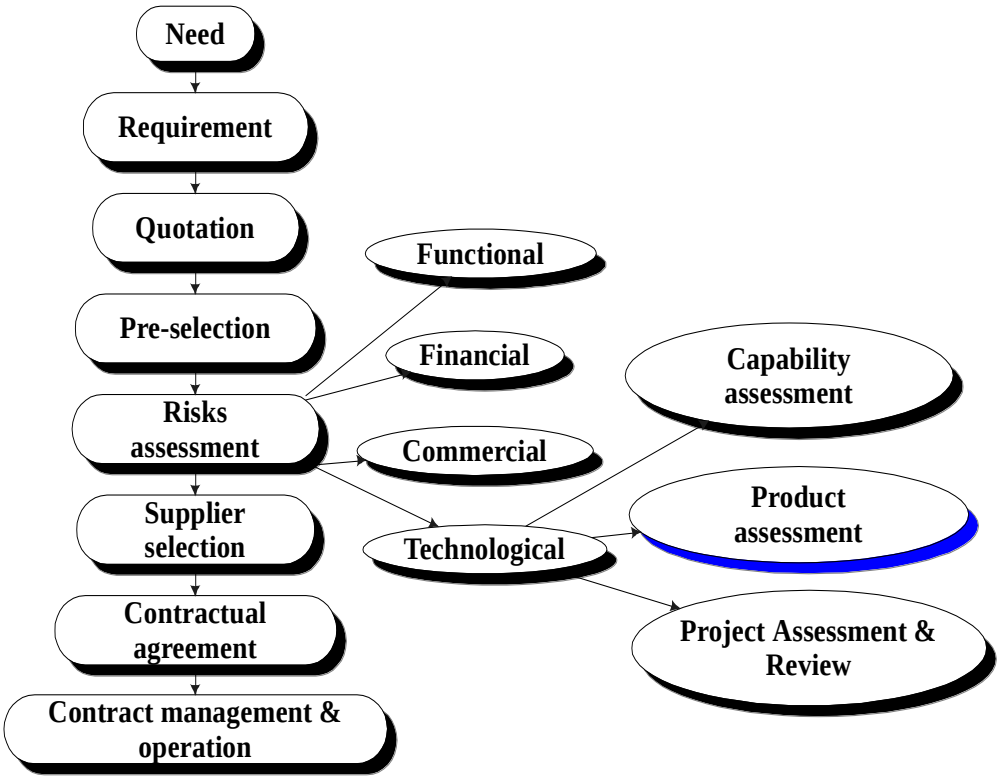


Figure 1: Acquisition Process and Risk Assessment

Risk factors.

Many risk factors must be assessed to insure a smooth procurement project.

These factors can be subdivided as follows:

- Process
- Project
- Product

Process capability factors are related to the practices, human resources and tools used by the supplier to develop and support its product. For instance, the absence of a quality system or proper testing techniques in a development organization is a significant contributor to risks. An improper estimation process will make schedules and resources estimates unreliable.

It is important to note that, while having a positive impact, an ISO 9001 certified quality system does not mean that a developer of software product has the necessary capability to be a low-risk supplier. Such a certification only means that, within a given geographical and organizational scope, the supplier is consistent in the process and practices that are applied.

Project factors are the constraints put on the development projects such as resources and schedule allocation, clarity and stability of requirements and mastery of the processes used for this development project. An excellent development organization can still miss schedule and quality objectives if given unrealistic objectives.

It is important to note that an organization's development processes are a subset of its overall business processes. Low maturity in some business practices will influence the performance of development projects, even if the development processes are very sophisticated.

Product factors are essentially related to the complexity and technical challenges related to the design of the product. Products whose development implies the simultaneous (or concurrent) development of silicon (e.g. IC's), hardware and software are inherently more risky.

While there is a large technical component in the above mentioned factors, organizational and human elements are important contributors. Most software development projects that get into trouble will be mostly because of improper management. Most technical and technological issues can and must be managed.

Technical Due Diligence

As mentioned earlier, our technical Due Diligence is performed doing three types of assessments. These are:

Capability assessment -	Capability assessments are done using, as a benchmark, our TRILLIUM model, a derivative of the Software Engineering Institute's Capability Maturity Model (CMM) described in Annex A. This model is a collection of "best in class" software product development and support practices. Such assessments help us in evaluating the development and support capability of a the supplier for a specific product line. These assessments are highly focused on the product being acquired.
Product assessment -	A product assessment is conducted if the product is critical or if the capability assessment indicates at least a medium or a high risk level. The product assessment aims principally to assess the maintainability of the software product as defined by the ISO-9126 standard. The product is thus assessed from an architectural, a design, and an implementation point of view. For those assessments, we go as far as doing a complete analysis of the source code in some cases. We have developed with a university a method and its associated tool that make such an assessment practical.
Project assessment -	A project assessment is done when the product is under development. These assessments are done to evaluate the probability that a given product will be delivered with all its defined functionality and within schedule. These assessments are conducted on a regular basis during the development life cycle of the product.

Risk factors assessment.

Assessing the risk factors mentioned previously is not a trivial undertaking. Not only does it require appropriate expertise (like for any type of Due Diligence), but it must be done under the schedule constraints of the procurement activity and the limitations associated with any auditing activities.

The most important resource needed to perform such a risk assessment activity is proper staffing. The personnel we use is an expert in either Management Information System (MIS) or telecommunication & software engineering with a graduate (Master) degree and at least 5 years of product development and auditing experience.

It is very important that the personnel we use be not only credible internally, but also for the vendors. Care must be taken in the selection of this personnel since on top of the technical experience it must have the managerial and interpersonal skills required from professional auditors.

A critical mass of such people is needed in a procurement organization to perform such a function. This is needed to insure the appropriate span of expertise and also peer review. Many assessments will need to be done by two experts to insure proper diagnostic and also to be able to do it in the short amount of time available on the vendor premises (One to two days usually). Also, proper resources must be available to maintain and expand the expertise of this personnel.

A risk assessment exercise is essentially an expert evaluation. While a systematic process can be tough and described for such a task, the actual implementation is not a "handbook engineering" process. The individual performing such assessment must:

- Be knowledgeable in the technology used in the product,
- Have the proper investigative skills,
- Have an excellent knowledge of the software (and system) engineering discipline.

The Trillium Model

The Trillium model [See Annex A] was developed by Bell Canada and Nortel. Our objective was to have a reference model that met the following criteria:

- Coverage of all practices that are necessary for the development and support of software products, specifically telecom products,
- A good synthesis of recognized benchmarks and standards for software products development and support,
- An architecture that make it easy to associate practices with specific discipline (e.g. Configuration management), and that give guidance on how such discipline should evolve and improve,
- A strong customer focus.

The resulting model was named Trillium (initially the name of a conference room where the team was meeting, in reality the name of a common flower of eastern Canada).

Trillium was intended to be a top level document for the CMM, ISO 9000-3 and the other standards that it is covering, not a replacement. Since a detailed two-way mapping between Trillium and the CMM as well as ISO 9000-3 and other documents exist, it is easy for users to go back to these documents and their supporting documentation for more details.

Pertinence of a CMM type model in a procurement risk management context.

As mentioned previously, risks assessment exercises are focused toward the identification of major risks factors associated with principally the delivery of a software products and its subsequent support.

A CMM type model, while a good reference, is at the same time too detailed and insufficient for such a task.

These models are too detailed since there is physically no time for assessors to go through all the practices. They are insufficient because they are limited to the software development process. Even the Trillium model with its wider scope is not sufficient. As mentioned previously, contextual elements (product complexity, architecture) as well as the overall maturity of the organization's business maturity must be looked at.

The model contributions are as a roadmap for information gathering and as a reference or benchmark for practices. What information to look at and in which sequence is most important when time is limited, and in a small organization like ours we pass this experience through on-the-job training. Interpreting this information properly and in context is a factor of the technical expertise of the assessor.

Conclusions.

Our processes and methods have been in constant evolution in the last three years, and we expect this to continue as we acquire more experience. Specifically, we must always fine tune our processes to adapt to the procurement of new type of products such as ATM and Broadband technologies or client-server based systems.

We have found that having such a capability in a purchasing organization is an asset as long as proper focus is maintained. The deliverable of a technical Due Diligence exercise is not a technical report but a diagnostic that can be used for decision making.

REFERENCES

Anonymous, 1991, ISO/IEC 9126, "*Information technology - Software product evaluation - Quality characteristics and guidelines for their use"*, ISO/IEC.

Anonymous, 1995, ISO/IEC 12207, "*Information technology - Software life cycle processes" "*, ISO/IEC.

Coallier F., 1994, "*How ISO 9001 Fits Into the Software World "*, IEEE Software January 1994 pp. 98-100.

Coallier F., McKenzie R.M., Wilson J.F. and Hatz J., 1995, "*TRILLIUM - Model for Telecom Product Development & Support Process Capability "*, Bell Canada (available on the World Wide Web at http://ricis.cl.uh.edu/trillium/).

Jones C., 1994, " *Assessment and Control of Software Risks "*, Yourdon Press computing series.

Mayrand J. and Coallier F., 1996, " *System Acquisition Based on Software Product Assessment "*, Proceedings of the 18th International Software Engineering Conference, Berlin, March 1996.

Paulk M.C., Curtis B., Chrissy M.B. and Weber C.V., 1993, "*Capability Maturity Model for Software ", version 1.1*, CMU/SEI-93-TR-24.

ANNEX A: Description of the TRILLIUM Model

Scope

The ***Trillium*** Model covers all aspects of the development life-cycle of a software product, most system and product development and support activities, and a significant number of related marketing activities.

Although ***Trillium*** has been designed to be applied to embedded software systems such as telecommunications systems, much of the model can be applied to other segments of the software industry such as Management Information Systems (MIS).

Many of the practices described in the model can be applied directly to hardware development.

Model Foundation

The ***Trillium*** Model is based on the Software Engineering Institute (SEI) Capability Maturity Model (CMM) version 1.1.

The architecture of the ***Trillium*** Model differs from the CMM version 1.1. The most significant differences are:

- a model architecture based on roadmaps, rather than key process areas,
- a product perspective, rather than software,
- wider coverage of capability impacting issues, and
- a customer focus, technological maturity, and a telecommunications orientation.

This version of the ***Trillium*** Model covers all SEI CMM v1.1 activities and abilities and some of the commitments, measurements and verifications (see Appendices for details of coverage).

In addition to the above, this version of the Model incorporates the intent of:

- ISO 9001: 1994 International Standard,
- ISO 9000-3: 1991 Guideline,
- relevant parts of the Malcolm Baldrige National Quality Award, 1995 Award Criteria,
- IEEE Software Engineering Standards Collection, 1993 Edition, and
- the IEC Standard Publication 300: 1984.

The ***Trillium*** Model incorporates additional practices from the following topics:

- Quality Management
- Business Process Engineering
- Technological Maturity
- Development Environment
- Systems Engineering
- Co-Engineering
- Concurrent Engineering
- Reliability Engineering
- Customer Support/Partnership
- Usability Engineering

The *Trillium* Scale

The ***Trillium*** scale spans levels 1 through 5. The levels can be characterized in the following way:

1. **Unstructured:** The development process is ad hoc. Projects frequently cannot meet quality or schedule targets. Success, while possible, is based on individuals rather than on organizational infrastructure. (Risk - High)
2. . **Repeatable and Project Oriented:** Individual project success is achieved through strong project management planning and control, with emphasis on requirements management, estimation techniques, and configuration management. (Risk - Medium)
3. . **Defined and Process Oriented:** Processes are defined and utilized at the organizational level, although project customization is still permitted. Processes are controlled and improved. ISO 9001 requirements such as training and internal process auditing are incorporated. (Risk - Low)
4. **Managed and Integrated:** Process instrumentation and analysis is used as a key mechanism for process improvement. Process change management and defect prevention programs are integrated into processes. CASE tools are integrated into processes. (Risk - Lower)
5. . **Fully Integrated:** Formal methodologies are extensively used. Organizational repositories for development history and process are utilized and effective. (Risk - Lowest)

Architecture of the *Trillium* Model

The ***Trillium*** Model consists of Capability Areas, Roadmaps and Practices.

Capability Areas

There are 8 Capability Areas within the ***Trillium*** model. Each Capability Area contains practices at multiple Trillium levels. For example, Management spans levels 2 to 4 while Quality System spans levels 2 to 5.

Roadmaps

Each *Capability Area* incorporates one or more *roadmaps*. A *roadmap* is a set of related practices that focus on an organizational area or need, or a specific element within the product development process. Each roadmap represents a significant capability for a software development organization.

Within a given *roadmap*, the level of the practices is based on their respective degree of maturity. The most fundamental practices are at a lower level whereas the most advanced ones are located at the higher level. An organization matures through the roadmap levels.

Lower level practices must be implemented and sustained for higher level practices to achieve maximum effectiveness.

The following table lists the roadmaps contained within each capability area, as well as the distribution of practices by maturity level and capability area.

Capability Area	Roadmap	Level Span
Organizational Processes Quality	Quality Management	2-4
	Business Process Engineering	2-3
Human Resource Development	Human Resource Development	
and Management	and Management	2-4
Process	Process Definition	2-4
	Technology Management	2-4
	Process Improvement & Engineering	2-5
	Measurements	2-4
Management	Project Management	2-4
	Subcontractor Management	2-3
	Customer-Supplier Relationship	2-3
	Requirements Management	2-4
	Estimation	2-3
Quality System	Quality System	2-5
Development Practices	Development Process	2-5
	Development Techniques	2-5
	Internal Documentation	2-4
	Verification & Validation	2-4
	Configuration Management	2-5
	Re-Use	2-5
	Reliability Management	2-4
Development Environment	Development Environment	2-5
Customer Support	Problem Response System	2-3
	Usability Engineering	2-4
	Life-Cycle Cost Modeling	2-3
	User Documentation	2-3
	Customer Engineering	2-3
	User Training	2-3

Relationship to other Standards

The following table provides a high level indication of the alignment of ***Trillium*** levels with key industry indicators and the standards which it encapsulates.

Key Industry Indicators	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4	LEVEL 5
Process	Ad-hoc	Project based	Organization-wide		
Standards	None	CMM Level 2+ IEEE(a)	CMM Level 3+ IEEE(b) ISO 9001 IEC 300 (c)	CMM Level 4+	CMM Level 5
Process Improvement	None	Unstructured	Deployed	Systematic	

(a) Stds. 730, 828, 830, 1016, 1028, 1058.1, 1063.
(b) Std. 1012

(c) as applicable to the hardware component of a system

Achieving level 3 on the ***Trillium*** scale means that an organization meets the intent of the following:

- SEI Level 3
- ISO 9001 (and the associated ISO 9000-3 Guidelines for Software),
- IEC 300 for system,
- IEEE Standards 730, 828, 830, 1012, 1016, 1028, 1058.1, 1063,
- the relevant parts of the Malcolm Baldrige National Quality Award Criteria, and
- additional ***Trillium*** practices not covered by these standards

Meeting the requirements of a ***Trillium*** practice does not necessarily imply meeting all the requirements of the corresponding referenced standards or documents.

> ***Note:*** The IEEE Software Engineering Standards referred to in the current version of **Trillium** are mostly oriented towards work products, e.g., software design description, project management plan. For the purpose of ***Trillium***, these are used as guidelines only.

How *Trillium* Practices are Developed

The set of practices in the ***Trillium*** model is built using the following algorithm:

1. Practices are taken from the SEI CMM Version 1.1.
2. ISO 9001 and ISO 9000-3 clauses are mapped to this set of practices and where possible, practices are modified to integrate these requirements.
3. All remaining ISO 9001 and ISO 9000-3 clauses (i.e., which could not be mapped) are added to the set of practices.
4. Bellcore standards clauses are mapped to the practices generated by steps 1, 2 & 3. Where possible practices are modified to integrate these requirements.
5. All remaining Bellcore standards clauses (i.e., which could not be mapped) are added to the set of practices.
6. The same process is repeated with relevant portions of the Malcolm Baldrige National Quality Award Criteria.
7. Practices from IEC 300 are added.
8. References to relevant IEEE standards are added.
9. ***Trillium*** specific practices are added to provide coverage of additional areas important to the telecommunications industry. These are based on professional benchmarks generated through the consensus of subject matter experts and validated in a peer review process.

When practices are extracted from the CMM, or other standards, they go through the following transformation, if applicable:

1. The practice is generalized by either removing references to "software", or replacing them by "product and services" or "systems".
2. The practice is generalized by either removing references to "development", or replacing them by "development and support".
3. References to "group" or other specific organizational units are replaced by "function".
4. Indirect references to specific documents are replaced by references to a process (e.g., "quality plan" by "quality planning"), or to "documentation" or "information".

Practices are assigned to a given level based on the following general guidelines.

- Practices that are considered fundamental for the successful conclusion of a development project are assigned to level 2.
- Practices that are considered to be organization-wide in scope or fundamental to the continuous improvement of the development process are assigned to level 3.
- Practices that deal with CASE technology or characterize advanced process maturity (e.g., change management, integration of defect prevention, statistical process control and advanced metrics) are generally assigned to level 4.
- Level 5 typically deals with advancing technology as it applies to process automation, formal methodologies and strategic utilization of organization repositories.

A model for technological transfer with respect to process improvement - A practical example: ASEC

Denis Roy, Applied Software Engineering Centre (ASEC)

ABSTRACT

As many governments around the world, the Canadian government considers information technology as a major driver for the economy. Seven years ago, the Canadian software community and the different levels of government realized that there was a need and inherent benefits in establishing a national technology centre focusing on software engineering.

As a result, the Applied Software Engineering Centre (ASEC) was created to assist the Canadian Software community in raising its capability in software engineering by providing access to the best possible software engineering solutions and related training.

ASEC's major goals were to initiate process improvement efforts, to accelerate necessary technology transfer activities and to promote technology insertion projects between governments, industry and universities.

Denis Roy, General Manager ASEC, will discuss the implications of launching such an endeavor as well as challenges encountered throughout the process. An overview of potential risks and concerns that were highlighted during the process and the solutions finally implemented as well as the lessons learned will also be included in the presentation.

A practical example: ASEC

When the concept of « Software Engineering Standards » became an issue some years ago, many organizations voiced the same types of concerns: WHY?; we know what we are doing; we have our own standards; this will increase our development cycle and so on.

A major concern at that time was whether the production costs would increase and if so, would it be possible to include them in the contracts.

Considering the « not as good as it was » economy, we find that a critical issue in today's market is the return on investment. We hear comments such as:

- Is this really necessary, (can we obtain a waiver...)?
- What are the costs involved?
- Will this open up new markets?
- Will this help to keep our company ahead of the competition?
- Will this increase our revenues?

When considering the Information Technology Industry, our approach to the return on investment issue is based on a « Mature Risk Management Approach ».

- How can we be sure that we are doing what our customer really expects on a continuous basis?

- How can we ensure that the system's various components developed by us and / or our partners will merge together?

- How can we guarantee that the « Critical System » we are developing will be used properly and according to user functionality specifications?

- Finally, how can we clearly demonstrate our capabilities and identify possible risks to our customers, to establish a fair risk sharing approach which is required for the success of any major development project?

A Fair Risk Sharing Approach between customer and contractor can only exist if a mutual, common and dynamic understanding of the problem, the solutions and, the process being used in its development and implementation can be developed.

This is WHY « Software Engineering Standard approaches » were developed. Ideally, standards stand for :

Collaboration: Collaboration between countries, organizations and individuals who are defining standards. Collaboration between organizations, groups and individuals who are developing a system.

Compatibility : Compatibility of requirements, approaches, processes, methods, techniques and system components.

Visibility: Visibility in requirements, risks, development processes and in product specifications and functionalities.

Recognition: Recognition of contractor capabilities and product certification.

Evolution: Evolution of standards based on software technology and application domain evolution.

Software project development standards such as 2167A (being replaced by 12207), quality system standards such as the ISO 9000 series, IEEE standards as well as models such as the SEI-CMM[sm] have now caught the attention of most senior managers within our software community. Standards are now recognized as competitive assets and discriminators in awarding contracts and, when correctly implemented, contributing to the profitability of the organization.

Therefore, as many governments around the world, both Canadian and Québec governments consider Information Technology a major driver for the economy. Seven years ago, the Canadian software community and both levels of government realized that there was a need as well as inherent benefits in establishing a national technology centre focusing on software engineering.

As a result, the Applied Software Engineering Centre (ASEC) was created to respond to an urgent need expressed by the Canadian industry, which is facing a pivotal challenge in today's market. While in most sectors, Information Technology has become a significant factor in productivity and innovation and that we can register a considerable increase in requests for software, the deficiencies existing in software development as well as the shortage of qualified personnel have seriously impaired the growth of the industry. We are no longer surprised that a project is over budget, that deadlines have been missed or at the lack of product reliability as well as system downtime due to software error. However, in some critical applications, these shortcomings can have catastrophic impact on public safety or generate important financial losses.

In 1988, a group of Canadian companies, including CAE Electronic, Canadair, Kéops informatique, Lockheed Martin, Oerlikon Aerospace and Spar Aerospace, who were more aware of the growing challenge, launched with the help of governments a feasibility study in view of establishing a centre who would ensure software engineering technology transfer to the industry and assist companies in increasing their capability level.

This study confirmed the role and the importance software engineering plays in improving productivity in the Canadian industry. Further to numerous consultations, this study measured the software engineering maturity level of the industry, as well as, comparing these findings with other countries and identifying the independent corporate needs. Throughout this process, the need for strategic alliances became very apparent.

As a result of the study, the sponsors decided to proceed with a business plan for the creation of a technology transfer centre who would be well versed in Information Technology and would assist the industry in improving their software engineering expertise.

The centre's mandate would be to ensure access to training on the best solutions available in software engineering be it technical as well as management.

Due to the size and urgency of the challenge, an association with an existing organization quickly became a priority in attaining the set objectives. The Centre de recherche informatique de Montréal (CRIM) was quickly identified as a partner who would be in a position to contribute to and support this new software engineering centre.

The Applied Software Engineering Centre (ASEC) was then created based on an agreement between the Centre de recherche informatique de Montréal (CRIM), Canadian Corporations who are active in the development of software for critical applications and the Canadian and provincial governments.

Further to an agreement signed on October 3 1991, the Applied Software Engineering Centre became a new division of the Centre de recherche informatique de Montréal (CRIM). Implementation and operational aspects were included in the agreement to ensure that ASEC would have the autonomy it required.

Its membership comprise companies and government agencies that rely on information technology to improve their productivity and the quality of their services and products, and that use complex software for critical applications.

Further to obtaining the government assistance required for its creation, ASEC has been in operation since September 1992. Federal funding in the amount of 2,8 Million dollars and provincial funding in the amount of 1 Million when added to independent corporate contributions and revenues generated by ASEC activities, represent a total budget of 7 Million dollars to cover a five year period (1992 through 1997). This budget has been applied to projects and services who assist Canadian companies in making necessary and radical changes in their software production process and to increase the quality and the reliability of products distributed on the global market.

The strategies used by ASEC in accomplishing its mission would be, first, to identify and promote applicable international standards and the best practices in software engineering; second, to accelerate the technology transfer process and to promote technology insertion projects creating links between government, industry and universities and finally, to supply technical expertise as well as strategic information to decision makers.

From the very start, potential commercial competitiveness between the Centre and member companies was subject to discussion. A unanimous decision was taken, at that time, so that ASEC should not be in competition with private companies when offering identical services. ASEC's mandate is to support the private sector in developing its capacity to meet new requirements.

This may seem like a simple task in theory. In reality, due to the type of requests for assistance received as well as the reach and the content of ASEC activities, it is difficult at times to clearly position services offered with respect to Canadian consultant agencies.

In addition, ASEC's mandate promotes the transfer of knowledge and expertise on software engineering standards and practices implemented, on an international basis, to various types of organizations be it integrators, users, developers and consultants, no matter the size and thus for the good of the community.

Also, within the scope of the agreement between ASEC and the Software Engineering Institute (SEI), ASEC's role is to transfer adequately and precisely the results of work in progress at the SEI to the Canadian community.

In the case where specific expertise, training or support required in adopting practices or international standards are not available within the private sector, ASEC supports the community by supplying the appropriate services. However, when the service becomes available commercially, ASEC must re-evaluate the content of its activities. Its technology transfer objective has then been reached for this application.

ASEC's services or activities are directed towards technology transfer and are focused on awareness, training and supporting organizations during the assessment process and the identification of required software engineering practices.

ASEC does not deal with generating, evaluating and implementing solutions, however, through specialized training, it focuses the organization so that it is in a position to identify and specify it's needs adequately, to define and implement appropriate solutions and, if required, evaluate proposed solutions. Therefore, ASEC assists in minimizing risks for clients as well as suppliers.

Software engineering being an area in full expansion, ASEC must constantly re-evaluate its reach as well as the content of its activities. In addition, if the transfer is performed efficiently, organizations will be more and more able to meet their own needs. It is therefore, not possible nor desirable to define fixed and final limits to ASEC activities. However, the discussions which allowed for these dynamic limits to be set will have to be repeated as required.

As a reference centre and an organization focused on technology transfer, ASEC must maintain a constant vigil throughout the international community in order to identify technologies more liable to improve practices of industries involved in the development and maintenance of software.

Information technology evolves at a fast pace and is under pressure to increase and add discipline to practices in order for the industry to become more competitive on the global market.

More and more individuals, groups or organizations of all kinds promote standards that are also becoming more and more numerous and varied. Unfortunately in some cases, more particularly with respect to quality, these promoters are not always aware of the full impact on the industry. This is even more problematic when dealing with the information technology community.

A partial knowledge of a standard or a quick interpretation can, at times, entail costly decisions for the community. In addition, we must be careful not to fall into the classic trap of simply implementing a standard because another country has indicated that their industry has conformed to it.

Standards are sometimes used as economic obstacles between countries. However, what would be the purpose of conforming to standards if there was no competitive advantage? Standards or more specifically quality standards, should ensure the client that a product meets his/her requirements as well as facilitating the suppliers role.

When discussing information technologies, imposing a standard unilaterally can constitute a risk for the industry. Even with the best intentions in mind, organizations identify, decide, and impose standards without consulting the industry. We can question the merits of such an approach.

In Québec, the need to conform to ISO 9001 quality standard system with the use of the application guide ISO 9000-3 for the software industry has become a reality. Camélia will also have an impact on our industry. In Canada, various groups including government agencies are pressing the industry to conform to their chosen standard or approach (ISO 9000, TickIT, 12207, etc...). Unfortunately, there is often confusion with respect to the guides, the models (ex. SEI CMM) and the standards.

ASEC plays a major role, at this time, in identifying, promoting and supporting the selection of international standards as well as the implementation of the best practices in software engineering in order to assist the Canadian industry in increasing its competitiveness and in maintaining or increasing its profitability. This in mind, we can see the importance of ASEC being a spokesperson for the Canadian Information Technology Industry with respect to standards. This can avoid obtaining certification without considering the collective objectives.

Generally, standards include an assessment process that leads to some form of certification without taking into account the return on investment of the practices implemented in order to attain this conformity. ASEC has taken a continuous improvement approach that, while assisting companies in improving, their capability, procures a better return on investment and also allowing to obtain the required « certifications » as a by-product of its improvement.

ASEC's mission is to help the Canadian software community raise its capability in software engineering, especially in critical applications, by providing access to the best possible software engineering solutions and

related training. More specifically, ASEC's strategy about STANDARDS is to identify, promote and support the implementation of APPLICABLE international standards.

Considering the evolution of standards and their impact on the Information Technology Industry, ASEC recently received, from its board, a specific, clear and strong mandate to represent and keep the industry informed about any developments and requirements related to standards within the international community.

In addition, ASEC also acts as spokesperson for the industry. Within the scope of its mission, and as decided by it's steering committee, ASEC has been mandated to intervene in all Canadian activities dealing with the identification, the promotion and the selection of standards as well as those imposed and the requirements for levels of conformity to the capability and process maturity models within the information technology domain.

ASEC faces major challenges in making information, expertise and international know-how more available resulting in a technology transfer even more profitable to the community it serves.

In 1995, the most important undertaking by ASEC was the international project to translate key materials into French. The Appraisal for Internal Process Improvement (CBA IPI) method developed by the Software Engineering Institute (SEI) is based on the Capacity Maturity Model (CMM) which is also developed by the SEI. Being an American institution, the products were available in English only. The translation of both the CMM and the CBA IPI fell under the direction of ASEC with the collaboration of the SEI and a proofing committee regrouping organizations from both France and Québec allowing to better serve the French speaking community.

Thus far, 1996 has also been a year of challenges. In fact, ASEC was a major player in setting up an international symposium: VISION 96 - Symposium on Process Improvement: Toward an International Vision. VISION 96 was organized by SPIN-Montréal (Software Process Improvement Network) in partnership with the Applied Software Engineering Centre (ASEC, Canada), The Software Engineering Institute (SEI, USA), the European Software Process Improvement Foundation (ESPIF, UK) and the Centre de recherche informatique de Montréal (CRIM, Canada).

Also in the context of its technology transfer mission, the Applied Software Engineering Center (ASEC) has developed a concise approach for identifying, mapping and evaluating risks associated with software processes. While based on the maturity model developed by the SEI, the S:PRIME approach is designed to meet the process metrology needs of organizations who have between 10 to 100 professionals.

In closing, ASEC is continuously searching for new ways of increasing its reach and in attaining its technology transfer objectives. Under development at this time, is an extension of the reach by way of videoconferencing and by offering training on-site at the Fort St-Jean Campus. These means reduce cost while maintaining the quality of services and products supplied by ASEC.

From the start, ASEC has recommended an approach based on a network of Centres and has established strong relationships with various provincial and regional organizations. In Canada and possibly everywhere, this approach is essential mainly due to regional industry specifics.

We prefer to have an international network of centres who are leaders in their own field rather than imposing a hierarchy in which the necessary latitude and recognition are not available. For the above reason, ASEC has developed strong relationships with key players in their community. Associations include an agreement with the Software Engineering Institute (SEI) in Pittsburgh USA, an international collaboration within the scope of the SPICE project as Regional Trials Coordinator for Canada and Latin-America, an agreement with INTEC CHILE (SPIN) in Chile for technology transfer and training as well as agreements with the « Conservatoire national des arts et métiers - CNAM » in France and Software Productivity Center (SPC) in western Canada as well as others.

In conclusion, the success of an Information Technology Transfer organization is based on its flexibility and in its capacity to bring about positive change to the software engineering community. Also, we believe that the worldwide information technology community is facing the same problems, that the solutions to these problems may be pretty similar, however, their implementation will inevitably defer due to industry culture specifics. In order to truly support the evolution of information technology, not only do we need to acknowledge cultural differences but we need to learn to cope with and accept them.

PICO - A New Strategy for Implementing the Paradigm of a Learning Organisation

The PICO Team
http://www.iol.ie/~iscn/projects/pico.html

Abstract:

PICO´s Mission is to develop a comprehensive set of configurable training courses packaged with a book as a basic information pool and with a tool for supporting automated generation of analysis data covering "Process Improvement from Analysis to Success".
PICO is based on the „learning by doing" training principle and on the paradigm of a learning organisation. Only those systems who are able to continuously adapt themselves to new situations and environments have a chance to survive. Learning and dealing with these new situations is a key success factor. Life that stops learning stops living.
This article outlines the different PICO components and products, the learning strategy, how the concept of a self learning organisation is supported, and will give organisations a detailed insight into PICO as a collaborative and learning based project.
The project is carried out with the financial support of the CEC under the Leonardo da Vinci Programme [11].

Introduction

A major requirement for any learning organisation is the ability to establish collaborative groupings and a culture in which partners are able to improve faster by working together than they could on their own [2], [8], [9]. One of ISCN´s goals is to act as an entrepreneur to establish project consortia among members to do projects together solving process improvement problems. The PICO consortium came together through ISCN and is a well selected group of highly acknowledged companies in the SPI field. It is a mixture of large companies, method providers, and trainers and experts as illustrated in Fig. 1.

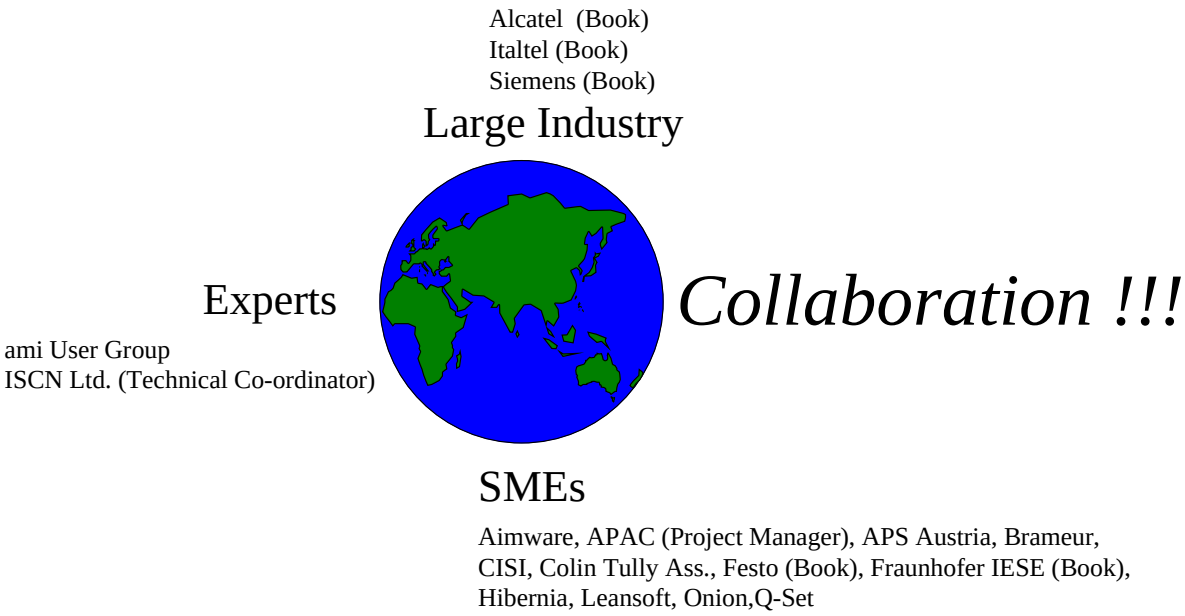


Fig. 1: The PICO Partnership - A Collaborative Grouping

The project runs in three major phases

- *1996* : Book Development and First Training Courses
- *1997* : Field Test and Feedback Evaluation
- *1998* : Final Industry Version and Large Scale Dissemination and Exploitation

It is expected that the field tests will start in late Spring 1997, the book will then be published.

Product Architecture

The PICO product set comprises (see Fig. 2)

- a book "Better Software Practice for Business Benefit" to which a group of large companies and SMEs contribute
- a set of configurable training courses and workshops covering process improvement from analysis to success
- an analysis tool for automated data visualisation and data collection support

The book

Major companies such as Siemens, Alcatel, Festo, Italtel, etc. method providers, and experts contribute their experience. The architecture of the book is divided into two parts: Part I Principles, Part II Practice. Part I starts from business factors, discusses software processes, presents different approaches for software process analysis, describes how to use a goal question based approach for business driven improvement planning, and outlines cost and ROI factors in process improvement. Part II represents a selection of case studies from different companies which relate to the principles described in Part I. The book framework has already been released, the authoring work is in process, and the book will be published in Spring 1997. The book will be both a handbook for managers and software practitioners as well as a reference material for the training courses.

The Training Courses

The workshop based training courses implement the „learning by doing principle". The following six courses are developed:

- each training course is a one day workshop
- the training courses are
    - Process Analysis Training
    - Goal Based Improvement Planning
    - Experience With Improvement Projects

  - Process and Product Measurement
  - Business Goals and Improvement Strategies
  - Self Assessment Tutorial
- a customer can select any combination of the one day modules
- each one day module can be separately held as a training course

Before starting the training course development a Course Architecture Guideline was designed to

- establish a framework for a set of integrated training courses
- provide guidelines for course development authors
- define a common architecture for the training courses

The course architecture guideline and all courses shall be generic which means that for future updates the architecture can remain the same, only the content of some sessions will be exchanged with more up-to-date case studies. This way the training course set becomes a framework for starting a process improvement learning initiative in large companies which can be adapted continuously (see also section 4).

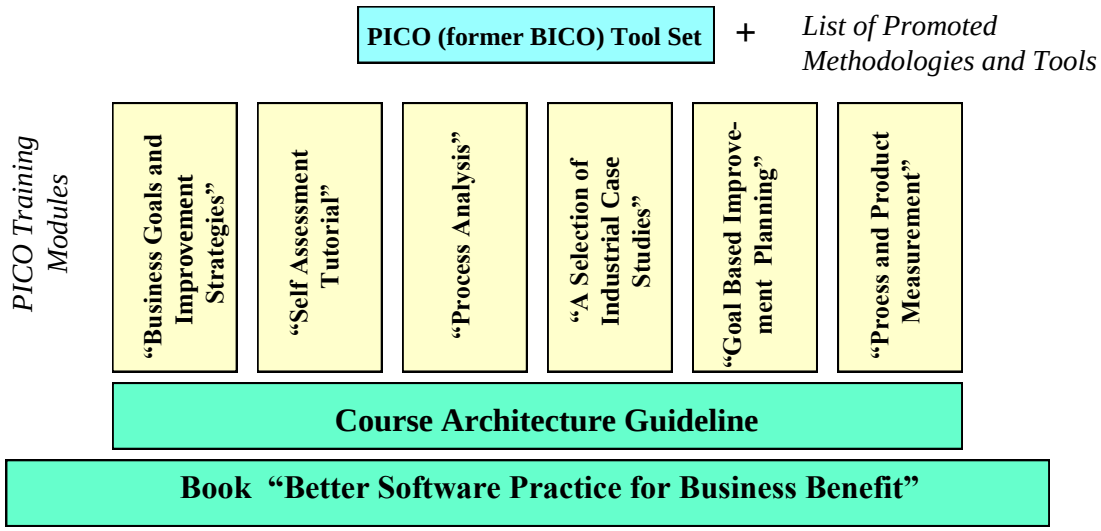


Fig. 2: The PICO Product Set

The analysis tool

The tool is based on an industry application which runs on a pocket calculator and has full functionality of a data collection and analysis tool. The tool is able to calculate maturity profiles for different configurations of process attributes. e.g. calculating a maturity profile for ISO 9001 attributes, or calculating a maturity profile for SPICE attributes. The main advantage of this tool is the fact that it runs with minimum resources and assessors can go through the organisation with a pocket computer in the jacket. At the moment the data collection component is available, the evaluation component is still in the development stage, a first version will be used in the field tests in 1997.

PICO promotes and develops further the tools designed in BICO "Benchmarking & ISO Combined". BICO is a tool supported quantitative method for assessments and improvement planning which is based on ISO 9001. It is based on an ISO 9001 conform questionnaire, a tool set that allows compilation and visualisation of data, and a formal approach to identify improvement potentials. One key strength is the fact that the tools run on a pocket size computer of HP.

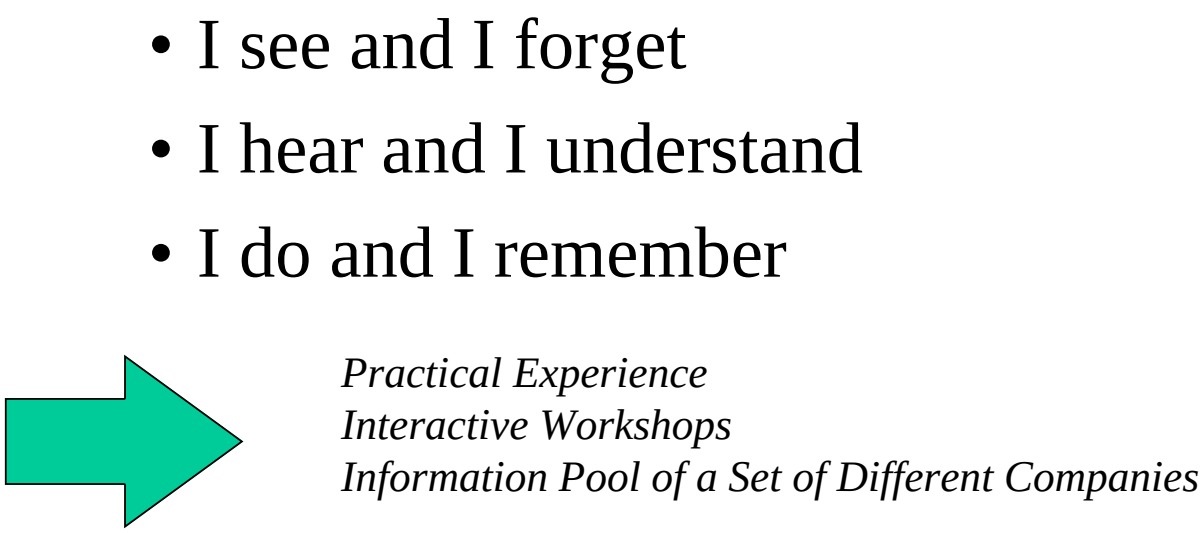


Fig. 3: The Learning Principle

The course development takes into account important psychological factors [1] illustrated in the above Fig. 3, active integration and involvement is the objective. For example, when this approach is applied to Project Management learning, then participants select a project that they have to manage in the coming months and use this as the case study for the workshop. This gives a few benefits:

- they apply the approach before leaving the class room environment and can see the practical problems and solutions
- they have their project somewhat started before they leave the class
- they see the approach as it applies to them and not some mythical company / project
- they are more active on the course and less likely to be bored!

Improvement Process Support

The entire set of training courses shall address all different levels within a company. The course on "Business Processes and SPI" shall address top managers, the courses on "Process Analysis" and "Goal Based Improvement Planning" shall address process improvement groups and middle management who have to employ the strategies of top management. And the courses like "Self Assessment Tutorial" and "Process and Product Measurement" and "Experience with Improvement Projects" shall address the project managers and practitioners who have to implement the improvement projects and to measure the results.
This way the PICO training course with its 6 modules shall cover all aspects to start and implement an improvement programme in an organisation. This is depicted graphically on the next page.

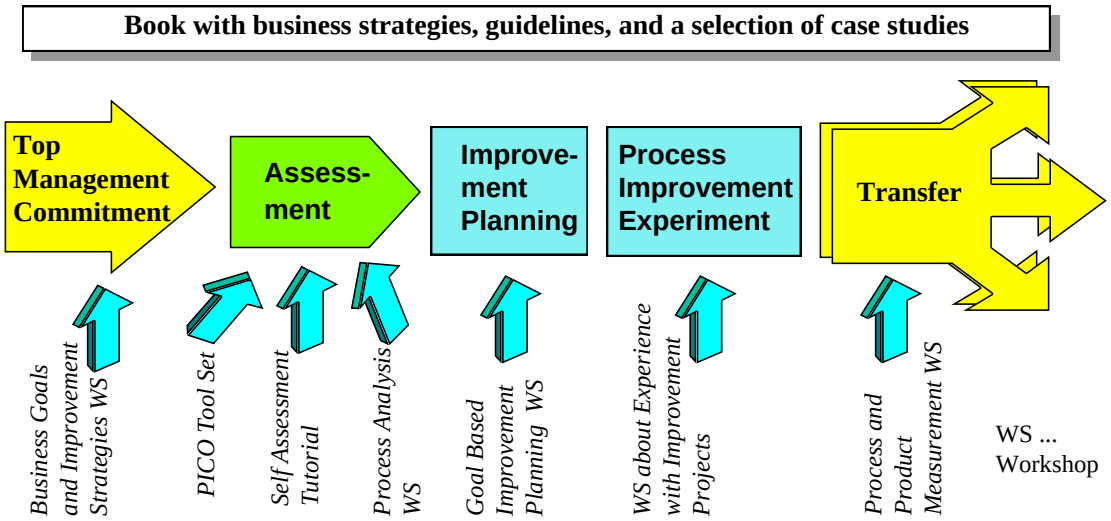


Fig. 4: Improvement Process Support by PICO Products and Workshops [10]

TRAINING MODULES	TARGET GROUPS	Language
• Business Goals and Improvement Strategies Workshop	Division Head, Company Head, IT Executives	Short and Significant Business Language
• Process Analysis Workshop • Goal Based Improvement Planning Workshop	Software Engineering Process Groups, Consultants, Improvement Teams	Manager´s Language
• Self Assessment Tutorial • Workshop About Experience With the Implementation of Improvements • Process and Product Measurement Workshop	Project Managers, Quality Managers, Practitioners	Practitioner´s Language

Fig. 5: Target Groups

It is a fact that business managers, project managers, and practitioners speak different languages and may have different viewpoints on the same situation [12]. Business managers speak about fixed cost, variable cost, return on investment, leveraging, market trends, product sales, and customer satisfaction. Middle and project managers speak about budget, work plans, quality plans, configuration management, requirements analysis and structured analysis, and are always in fear of delayment or exceeding the budget provided by the business managers. Practitioners deal with modules, design them, implement and test and deliver them so that they can be integrated into the system architecture planned by the project manager. It is the nature of process improvement methodologies that measurement and control functions are installed which again will be seen differently from the different target groups (see Fig. 6). Business managers not understanding that SPI needs investment with a ROI in about 3 years sometimes demand that process improvement is performed without any assignment of budget to it: lets do quality but it should not cost any dollar. This attitude leads to disaster and top management

commitment is the number one success criteria for starting an improvement programme. Middle managers will be most enthusiastic towards the process improvement because it provides them with methodologies and facilities to better define the processes, to better visualise the productivity and quality, and to improve the predictability which leads to the fact that schedules and budgets are kept according to the satisfaction of the business managers. At the beginning the practitioners usually see the implementation of a process improvement programme as a "dirty trick" of middle and project management to better control their performance. However, after some time they start to realise that more reliable plans give them enough time for design. Better design reduces the re-work and maintenance stress, formalised reports help them to identify the root cause of problems and to track the correction, and they can learn and improve themselves based on these measures [7].

Observed Reasons for Schedule Slips and Budget Overruns				
Rank by Business Managers (BM)	Rank by Project Managers (PM)	Rating (1-5) 1 .. rarely 2 .. sometimes 3 .. often 4 .. likely 5 .. always	Reason	Agreement
1	10	BM = 4,25 PM = 2,5	Insufficient Front-End Planning	Disagree
2	3	BM = 4,05 PM = 3,75	Unrealistic Project Plan	Agree
3	8	BM = 3,85 PM = 2,75	Project Scope Underestimated	Disagree
4	1	BM = 3,65 PM = 4,2	Customer/Management Changes	Disagree
5	14	BM = 3,1 PM = 1,7	Insufficient Contingency Planning	Disagree
6	13	BM = 3 PM = 2	Inability to Track Progress	Disagree
7	5	BM = 3,4 PM = 3,3	Inability to Detect Problems Early	Agree
8	9	BM = 3,1 PM = 3,2	Insufficient Number of Checkpoints	Agree
9	4	BM = 2 PM = 3,75	Staffing Problems	Disagree
10	2	BM = 1,9 PM = 3,75	Technical Complexities	Disagree

Fig. 6: Project Managers´ and Business Managers´ Different Viewpoints [12] 1988

As may be seen from Fig. 6 any improvement programme at the beginning is confronted with a number of different viewpoints and conflicts. Thus, it is mandatory to start any improvement programme with an objective and quantitative analysis which allows the different target groups to focus on the improvement of certain weaknesses. It is of critical importance to produce a training product which allows all these different viewpoints and target groups to be addressed and creates a joint vision for improvement within a software organisation.

Therefore any restriction to only one target group leads to a situation where the training product cannot be used for all phases from problem awareness, business goal planning, analysis, to success. For this reason within PICO it was decided to develop a number of workshops and products that support the different target groups in all phases of the improvement process (see Fig. 5).
A study by Barry Boehm, for instance, illustrated that personnel capabilities have at least a two times higher impact on productivity than any other factor. In addition to this Watts Humphrey recently developed the Personal Software Process [7] which emphasises that a process focused organisation runs best if the individuals themselves act as defined processes measuring their effort, failure rate, and productivity.

The Concept of a Self Learning Organisation

A new philosophical paradigm (the radical constructivism) defines the world as a dynamic network of processes which highly interact with each other. Humans are seen as part of this network, the human brain itself acting as a process. And like human beings every process in the world underlies a continuous evolution.
This means that change is a natural requirement of a never-stopping evolution of the world [6]. And any evolution of a process leads to an adaptation of its interfaces to all other processes. So it is wrong to believe that a stable and never-changing software system can be achieved. The development and management of software systems must concentrate rather on the adaptability of software systems and change management as a key organisational process.

The innovation cycle in industry is becoming shorter and shorter. After delivering a product version to a customer the following things may change

- the customer requirements (additional functions needed, change in organisational environment)
- the market demand
- the underlying technology platform
- the system and operational environment
- product changes due to problems
- the underlying software environment (new versions from Microsoft appear every 6 months)
- the management requirements (ISO 9001, subcontractor capabilities)
- the development team changes
- business factors (a developing company stops further development because revenue can be made by other services and products)
- the politics (e.g. in the defence sector the budget is reduced by half)
- etc.

As Darwin had already outlined over a hundred years ago, only those species who continuously adapt themselves to new situations and environments will survive. The same principle can be applied for software organisations and markets. (see Fig. 7)

- Continuous Evolution
  - underlying any system, process, human being etc. is a continuous evolution as a natural principle
  - only those (ref. Darwin) who are able to adapat themselves continuously to the changes caused by the evolution will survive
  - this requires continuously learning to manage new situations and change
  - In biology this principle has already been fully acknowledged
- Self Learning Organisation
  - Darwin´s principle cannot only be applied to living things
  - Darwin´s principle can be applied as well to software organisations which have to continuously adapt to new business and market situations, customer requirements, quality requirements, etc.
  - Therefore: only those software organisations who are able to continuously adapt themselves to new situations will survive

Fig. 7: The Self Learning Organisation Principle

PICO supports this concept of a „Self Learning Organisation“ by

1. providing a set of courses and materials which train all different target groups (from business managers to practitioners) in a software organisation for starting, planning, and implementing a continuous improvement programme which provides the organisations with capabilities for change management, and by
2. establishing a course architecture guideline as a framework of such courses which allows the courses to be adapted within this framework continuously without changing the general course architecture.

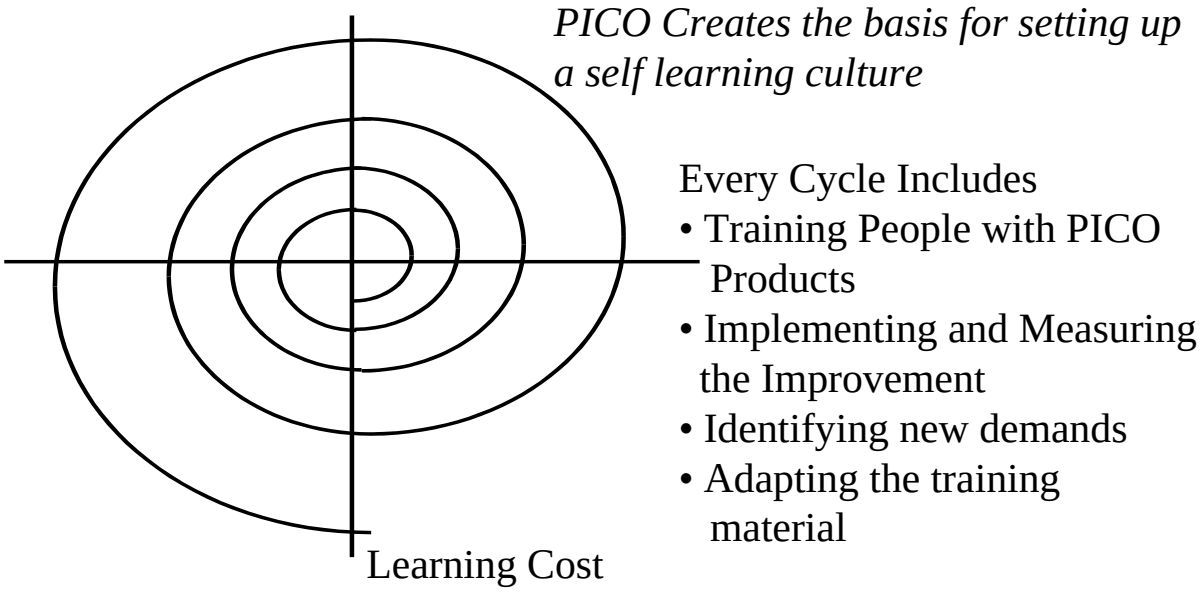


Fig. 8: The Evolutionary Learning Curve

Point 2. above allows large companies to use the entire PICO product set as a key learning initiative within their organisation. Firstly the PICO experts train trainers of the organisation who then perform the workshops within their organisation. Due to the workshop based architecture of the training modules every workshop will lead to new examples and experience which can be included in the next training. This way PICO only starts a Spiral Model Learning Initiative within large companies.

Variety and Diversity

PICO´s banner (as well as ISCN´s) is variety and diversity. The project is not restricted to only one methodology, it rather emphasises the combined use of improvement approaches and methodologies. The book will especially have a very broad scope and will illustrate different industrial case studies using different sets of methodologies to achieve higher quality, productivity and customer satisfaction.
Typical examples are the book contributions from Italtel and Alcatel, both dealing with the establishment of a long lasting improvement programme for Telecom switching systems with significant results and both using different methodologies to achieve it.

Collaboration and Partnerships

Roles of Project Partners

AIMware (Ireland)
- Training Course Development:
- Course Architecture Guideline
- Book and Integration Guideline:
- Writes an Experience Chapter for the Book

AMI User Group (Belgium)
- Book and Integration Guideline:
- How to Plan Process Improvement
- Training Course Development:
- Goal Based Improvement Planning

APAC (Austrian Product Assurance Company)
- Project Management:
- Project Leader, Quality Management, Configuration Management
- Training Course Development:
- Process Analysis Training
- Material Development:
- Further Development of BICO Tool

APS Leonardo (Austria)
- Dissemination:
- Field Test Organisation

Brameur UK Ltd. (UK)
Training Course Development:
Process and Product Measurement

CISI (France)
Training Course Development:
Field Testing, Review, and Adaptation to French
Book and Integration Guideline
Writes an Experience Chapter for the Book

Hibernia Learning Partnership (Ireland)
Dissemination:
Field Test Organisation, Training of Trainers

ISCN (The International Software Collaborative Network)
Technical Review and Co-ordination
Book and Integration Guideline:
Book Editing, Process Model Chapter, Planning and Managing the Book Development
Training Course Development:
Business Goals and Improvement Strategies
Dissemination:
Software Process Congresses, Tutorials, Leaflet, WWW

Leansoft Oy (Finland)
Training Course Development:
Self Assessment Tutorial

Onion Communications (Italy)
Book and Integration Guideline:
Process and Product Measurement

Q-Set Ltd. (Ireland)
Training Course Development:
Experience with Improvement Projects

Contact Points

Project Director

Hans Scherzer
APAC
Mariahilferstr. 89a
A-1060 Vienna
Austria
Tel.: +43 1 663 806950
Fax.: +43 1 586 1284
email: apacshz@ping.at

Technical Director

Dr. Techn Richard Messnarz
ISCN Ltd.
Co-ordination office
Florence House, 1 Florence Villas
Bray, CO Wicklow Ireland
Tel.: 353 1 286 1583
Fax: 353 1 286 5078
email: iscn@genrix.ie

Additional Book Contributors

- Alcatel
- Colin Tully Ass.
- Festo
- Fraunhofer IESE
- Italtel
- K&M Technologies
- Siemens
- MTA SZTAKI

Potential for Co-operation

It is a declared strategy of this project to be based on industry demands and to establish training courses focusing on learning by doing and practical experience. Please give us feedback if you are interested in the results of this initiative, if you plan to co-operate with PICO as a field test partner (e.g. doing PICO trainings in-house in 1997), or if you feel that you could provide us with important requirements that shall be taken into account by PICO.

Conclusions

PICO is not competing with any of the existing methodologies. The PICO product set can be seen as a training course set which helps business managers to understand the need for process improvement, to support the middle and project management to select a proper set of improvement methodologies, and to help practitioners to understand (from practical examples) how to implement the improvement projects. PICO will establish and include a list of promoted methodologies and the architecture will allow the insertion of further methodologies and experience without changing the underlying course framework.
The PICO partners do not plan to found an EEIG after the project, bureaucracy shall be avoided as much as possible. The major goal is to field test and to spread the trainings across Europe as much as possible.

References

[1] Andersen W.L., *Technology Transfer is a Social Phenomenon*, in (eds.) Przybylinsky S., Fowler P.J., *Transferring Software Engineering Tool Technology*, pp. 56-57, IEEE Computer Society Press, USA 1987

[2] Biro M., Kugler H-J., Messnarz R., et. al., BOOTSTRAP and ISCN - A Current Look at a European Software Quality Network, in: *The Challenge of Networking, Proceedings of the CON´93 Conference*, Oldenbourg, 1993

[3] Boehm B.W., A Spiral Model of Software Development and Enhancement, in: (eds.) Nahouraii E., Butler J.T. Kavi K., Petry F.E., Richter C., Tham K., *Software Engineering Project Management*, pp. 128-142, Comput. Soc. Press of the IEEE, Washington, DC, USA 1988

[4] Dewayne E. Perry, Staudenmayer N.A., Votta L.G. *People, Organisations and Process Improvement*, pp. 36-45, IEEE Software, July 1994

[5] Engels G., Groenewegen L., SOCCA: Specifications of Coordinated and Cooperative Activities, in: Finkelstein A., Kramer J., Nuseibeh B. (Eds.), *Software Process Technology*, John Wiley & Sons, pp. 71-100, 1994

[6 Gaarder J., *Sophie´s World - An Adventure in Philosophy*, Phoenix House, London, 1995

[7] Humphrey W.S., The Personal Software Process in Software Engineering, in: *Proceedings of the Third International Conference on the Software Process*, pp. 69-77, October 1994

[8] Jarczyck A., Corporate Research and Development, Siemens AG, Collaborative Software Engineering - Information Sharing in Collaborative Environments, in: *IEEE Third Workshop on Enabling Technologies - Infrastructure for Collaborative Enterprises*, IEEE Computer Society Press, 1994

[9] Messnarz R., Mac an Airchinnigh M., Decrinis P., Felmann R., Penney D., ISCN - Managing Expert Networks, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[10] Rohen M., ESSI - The European Systems and Software Initiative, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

[11] Stewart J., Leonardo - The Networking Programme, in: *Proceedings of the ESI-ISCN´95 Conference on Practical Improvement of Software Processes and Products in September 1995 in Vienna*, ISCN Ltd. Dublin, Ireland, 1995

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

[13] Zawacki R.A., How to Pick Eagles, in: (eds.) Nahouraii E., Butler J.T. Kavi K., Petry F.E., Richter C., Tham K., *Software Engineering Project Management*, pp. 128-142, Comput. Soc. Press of the IEEE, Washington, DC, USA 1988

Improving total software performance in the context of total business performance

Colin Tully

Colin Tully Associates (CTA)
and International Software Collaborative Network (ISCN)

Abstract The predominant paradigm for improving software product quality is to improve the software process, based on the assertion that "the quality of a software product is largely determined by the quality of the process used to develop and maintain it". In turn, the paradigm depends on maturity models, against which process quality is measured both before and after improvement. Adopting a systems viewpoint, in which software production is treated as a complex system and modelled in terms of both elements and relationships, the paper examines how process maturity models may affect product quality, and how process maturity may interact with other causal factors.

Process maturity models: an introductory review

For the last five years or so, organisations undertaking software (and systems) development have increasingly shown interest in process maturity models, as a useful basis for appraising and improving their software (and systems) processes. The best known process maturity model is the SEI CMM for Software. It has spawned many derivatives, in software, systems and other domains. Some of those derivatives call themselves CMMs (capability maturity models), while others (such as Bootstrap, Trillium, Software Technology Diagnostic and SPICE) do not. In this paper the neutral non–proprietary term "process maturity model" will be used to denote any member of the whole class of such models.

From the outset, it is worth being clear in our minds about what a process maturity model is, and what it is used for. As precisely suggested by the name, it is a model of process maturity; its use is as a standard for measuring the maturity of a process. Similarly, the calendar and clock form a model of time, used as a standard for measuring time — though the similarities between those two modelling and measurement systems may be outweighed by more substantial differences. Perhaps a better comparison might be with the Michelin model for hotels and restaurants, used as a standard for measuring their quality.

A process maturity model is not the same thing as a process model, which is (as again precisely suggested by the name) a model of a process or a class of processes. Note, however, that process maturity models *contain* process models. They also *require* process models. The process models which they *contain* are of a different kind from those which they *require*.

Process maturity models *contain* process models because their strategy is first to propose a standard hierarchical decomposition of a process, which is a complex system; and second to propose a means of appraising that complex system by measuring its component parts, and of then aggregating the separate measurements. (That follows the reductionist approach of classical science and engineering.) The standard hierarchical decomposition, which is embedded in any particular process maturity model, constitutes a process model which is simple, static and general. We will call process models of this class *general process models*.

Process maturity models *require* process models because an important characteristic of maturity for any

process is that it should be defined (in other words, some kind of process model, more or less formal, should exist). Since having a defined process implies having a definition process, all process maturity models include the definition process as a component of the total process, though they offer no prescription about what kinds of process model may be used for the definition. We are now talking about process models which are less simple, less static and less general than general process models, and which by contrast describe real processes in a real organisation. We will call process models of this class *organisational process models*.

A systems view of process maturity models

The previous section reviewed familiar concepts, though possibly in a slightly unfamiliar way. The purpose of this section is to consider the organisational context in which process maturity models, general process models and organisational process models are used. That context will be set out in the form of a simple model.

First of all, however, it is necessary to introduce three more concepts relating to an organisation. They are *software product portfolio, actual process* and *imagined process.*

The *software product portfolio* is the set of software products which have been (or are currently being) produced and/or procured by an organisation. They may be for use by the organisation itself, or by clients, or both. If **SP** denotes the software product portfolio, $\mathbf{sp_i}$ denotes each individual software product, and **{x}** denotes "set of all x", then $\mathbf{SP = \{sp_i\}}$.

The SP is the only *raison d'être* for a concern with the software process, and for all the paraphernalia of models discussed above. Their use and value is solely to improve the fitness for purpose of the SP. A slight alteration yields the classical formulation: "The underlying premise of software process management is that the quality of a software product is largely determined by the quality of the process used to develop and maintain it" [Paulk 1995]. The model in this section, and indeed the whole of the remainder of this paper, is an attempt to investigate the extent to which that formula matches reality. The questions investigated are "how largely?" and "by what means?".

The *actual process* is the set of real–life processes which have been (or are currently being) used to produce and/or procure (and to maintain) the SP. These are human decision processes, undertaken by people controlling the use of methods and tools. If **AP** denotes the set of all actual processes, $\mathbf{ap_i}$ denotes the set of processes for an individual software product, and **U(x)** denotes "union of all x", then $\mathbf{AP = U(ap_i)}$. Note that AP will contain conflicting processes, which achieve a common goal in different ways for different products.

If the Paulk formula had unambiguously referred to "the actual process", then it could be accepted without hesitation; indeed not only would it be true, it would even be too weak. The phrase "largely determined" could be replaced by "entirely determined". The human decisions used in producing and procuring a software product, together with the methods and tools which are used in making them or triggered by them, wholly determine the product: there are no other causal factors. But since the critically important distinction is never made between the actual process and models of that process, the statement in its ambiguity can only stand as it is, and we are left asking "how largely?".

The *imagined process* is the set of individual perceptions which people have about the actual process. These are mental models, held by people who participate in, influence, or just know about, the actual process. If **IP** denotes the set of all imagined processes, $\mathbf{ip_i}$ denotes the set of processes as imagined by any one individual, and **U(x)** denotes "union of all x", then $\mathbf{IP = U(ip_i)}$. Note that each ip_i represents only a partial view, and that IP will contain conflicting perceptions of the same process.

Figure 1, below, shows the elements of the system which we have just described.

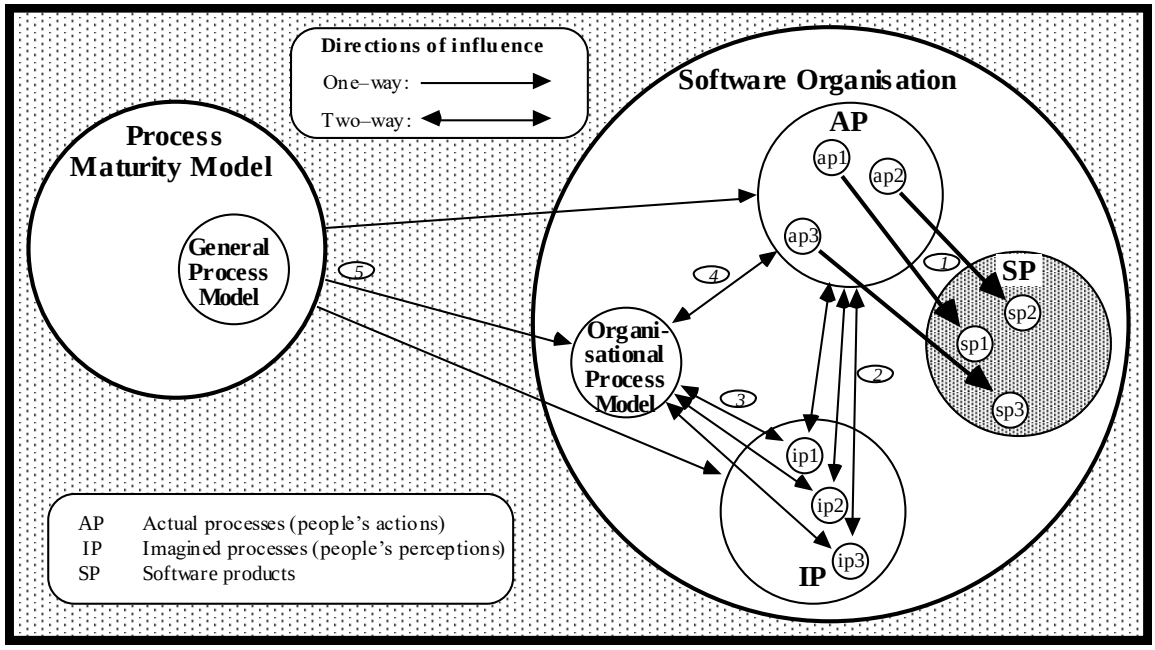


Figure 1 Process models, processes (actual and imagined) and products

The purpose of this section is to present a systems view of process maturity models and their context. A system is not just a set of elements but also a set of relationships among those elements. Figure 1, therefore, shows not just the elements that have so far been described, but also some relationships among them. The relationships shown are confined to relationships of a single type, which may be called the "influences" relationship. The instances of that relationship, and the direction of influence, are shown by arrows. Five groups of influences are shown, labelled from 1 to 5. Each of the five groups will now be discussed in turn.

Group 1 are influences from actual processes to software products. This is the clear and strong relationship which has already been discussed, and could be expressed in the form "the quality of a software product is *entirely* determined by the quality of the *actual* process used to develop and maintain it". The clarity of the relationship arises because it is one–to–one: each ap_i determines one and one only sp_i. The fact that this is the strongest of the groups of relationships is indicated by the boldness of the arrows.

Group 2 are influences between actual and imagined processes. They are two–way influences: the actual processes influence the imagined ones, and *vice versa*. For simplicity, the figure shows many–to–one relationships between individual ip_i and AP (the union of ap_i). Showing the reality, of many–to–many relationships between individual ip_i and individual ap_i, would excessively complicate the figure.

Group 3 are influences between the organisational process model (assuming that it exists in any documented form) and the imagined processes. These are again two–way influences: the imagined processes are important determinants of the organisational process model, and it in turn affects individuals' perceptions of process.

Group 4 are influences, again two–way, between the organisational process model and the actual process. Once more, for simplicity, the figure shows the relationship applying to AP (the union of ap_i) rather than to individual ap_i.

Finally, *group 5* are the influences between an external, public–domain process maturity model, with its embedded general process model, on (a) the actual process AP, (b) the imagined process IP and (c) the organisational process model. These are one–way influences: it is improbable that an individual software organisation will be able to have an impact on a process maturity model. Any impact it can

have will be mediated through the general user community for the maturity model, and will accordingly be slow, uncertain and diluted.

From a systems viewpoint, two key observations can be offered on the simple first–order model in figure 1.

- The most obvious feature is the bi–directional loop at the centre of the figure. It is a feedback loop linking AP, IP and the organisational process model, where each one can influence each other, and the influence can in turn be passed on round the loop. Such causal feedback loops are extremely important drivers of change in complex systems, and a great deal depends on their characteristics. For instance, what is the speed at which influences are propagated round the loop? Does the loop exercise positive feedback (to amplify the rate of change) or negative feedback (to dampen the rate of change)? Are the changes that are amplified or dampened desirable or undesirable ones? How do the various influences that propagate around the loop, in either direction, affect one another? The answers to such questions are, of course, highly specific to individual organisations: they can vary widely, giving rise to a wide range of improvement behaviours. Because of the complexities of the bi–directional loop, those behaviours may often be unpredictable and unstable — in the terminology of CMM level 1, chaotic! (Note that the presence of an organisational process model, as prescribed for CMM level 3, is not in itself guaranteed to remove the chaos. A systems theorist would recognise that, while it may well be a necessity for improvement, its ill–managed introduction could make things worse rather than better.)
- Outside that loop, there is no other feedback. Software products (SP) have at best a weak influence on actual processes (AP), and the experience of the organisation as captured in in its own process model has at best a weak influence on process maturity models in the public domain. (There can, of course, be exceptional cases where these feedback influences are not weak.) The dangers of such a state of affairs should be fairly obvious. One obvious danger, for instance, lies in the thoughtless application of an external process maturity model, which is ill–adapted to the realities of an actual organisation.

Thoughtful consideration of figure 1 also generates two important questions, both addressed in the next section.

- If AP determines SP, and if the process–related influences on AP (IP and the organisational process model) may have unpredictable and unstable effects, are there other influences on AP, of which we should take account, and what are they?
- What else, other than SP, is determined or influenced by AP (and by the other influences revealed in the answer to the first question)?

Other influences on the actual process: factors of production

This section introduces the term “factors of production” and applies it to software.

In classical economic theory there were four factors of production: *land*, *labour*, *capital* and *entrepreneurship*. Economists were concerned with them in order to be able to explain the levels of the returns earned by those factors of production, which were (respectively) rent, wages, interest and profit. Those returns constituted the costs of production, and were therefore the essential focus of supply–side economics (in contrast to the demand– side study of prices).

In the latter half of the twentieth century, popular management theory has also identified four factors of production, but different ones. Following the style of presentation for management, they all start with the same letter, so that they can be easily remembered by busy executives with little time (and perhaps inclination) to think. They are: *men*, *machines*, *materials* and *money*. They are used as a way of organising the description and teaching of management activities, because it is assumed that they are the subject matter of all management decisions.

We are now entering what, with different emphases, may be called the post–industrial society, the global information society, or the third wave. Software is clearly a major product category in this new era, and one which earlier notions of factors of production do not fit. Conventional wisdom identifies three factors of production for software: *process*, *people* and *technology*. Figure 2 shows a version of that current

wisdom.

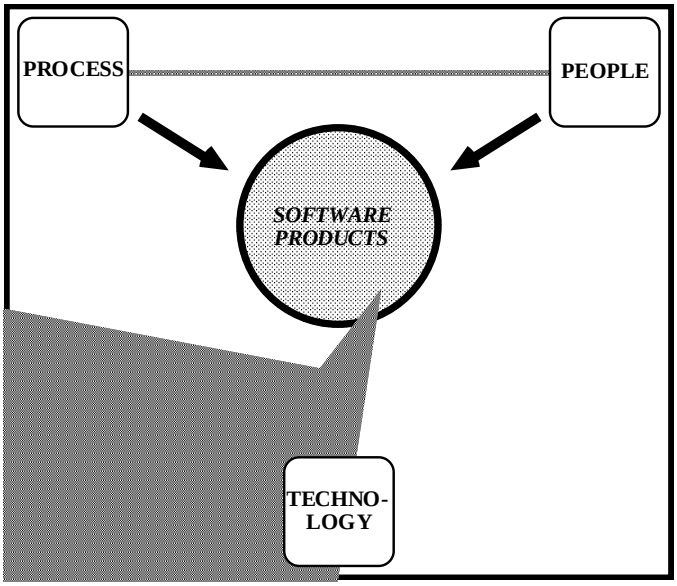


Figure 2 The conventional "factors of production" for software

They are, of course, not often called factors of production, because there are few software engineers with a background in economics, to whom the term might come naturally. Their purpose, though, is similar to that for the four factors in popular management theory: they are used as a way of organising the description and teaching of software quality and process management, because it is assumed that they are the subject matter of decisions made in that domain.

The triangle may be drawn with its apex at the top or the bottom, and with the three factors variously distributed among the vertices. It is drawn in figure 2 in the way it is, so as to emphasise the subsidiary and supporting role of technology, and the dominant importance of process and people.

The edges of the triangle, however, emphasise that the three factors do not operate independently of each other. We have a system of interacting factors. They can, however, be varied independently. They usually are varied independently, in fact, without consideration for the systemic interactions among the factors.

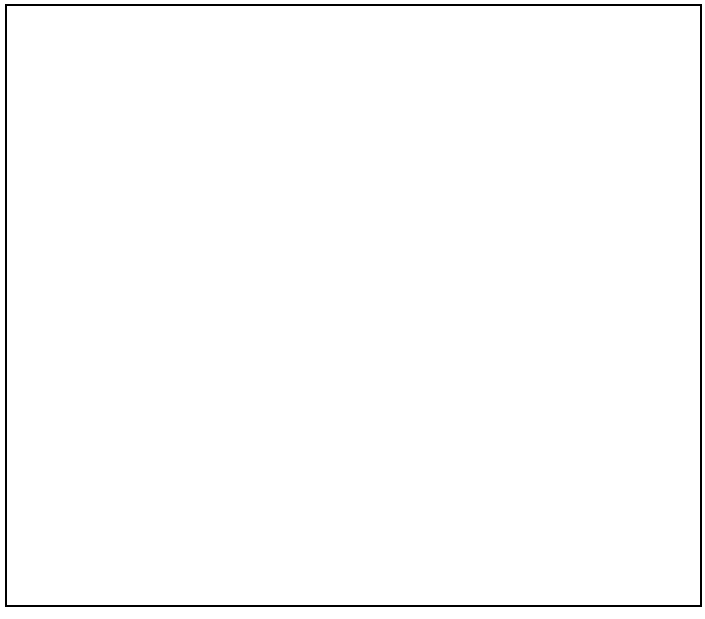


Figure 3 The conventional "factors of production" for software, modified

Two adjustments to figure 2 adapt the conventional wisdom to take account of the discussion in the preceding section of this paper. First, the central circle is relabelled "the actual process producing software products". Second, the process vertex is relabelled "established process", where the established process is the union of the imagined processes and the organisational process model (from figure 1). These changes yield figure 3.

It is now necessary to propose, however, that just three factors of production of software, as shown in the above models, are not enough. Consideration of wider–ranging models of product/service quality, such as the Malcolm Baldrige National Quality Award [Reimann 1993] and the European Quality Award [Waldner 1995] suggest the need for considerable extensions. The following changes are proposed to the model in figure 3.

- Factors contextual to the software organisation clearly impact its operation. These may be partitioned into factors internal to the parent organisation, which we will call *organisational factors*, and those external to it, which we will call *external factors*.
- Resources available for deployment by the software organisation also impact it. These may be partitioned into *information resources* and *financial resources*.
- Technology may appropriately be partitioned into *methods* ("soft technology") and *tools, platforms etc* ("hard technology"). [Methods and tools can be varied independently, and can therefore usefully be regarded as distinct. For some purposes, methods are regarded as part of process; but again the two can be varied independently, and can therefore usefully be regarded as distinct.]

Those changes yield eight main factors of production in place of the conventional three, enabling us to replace figure 3 with figure 4.

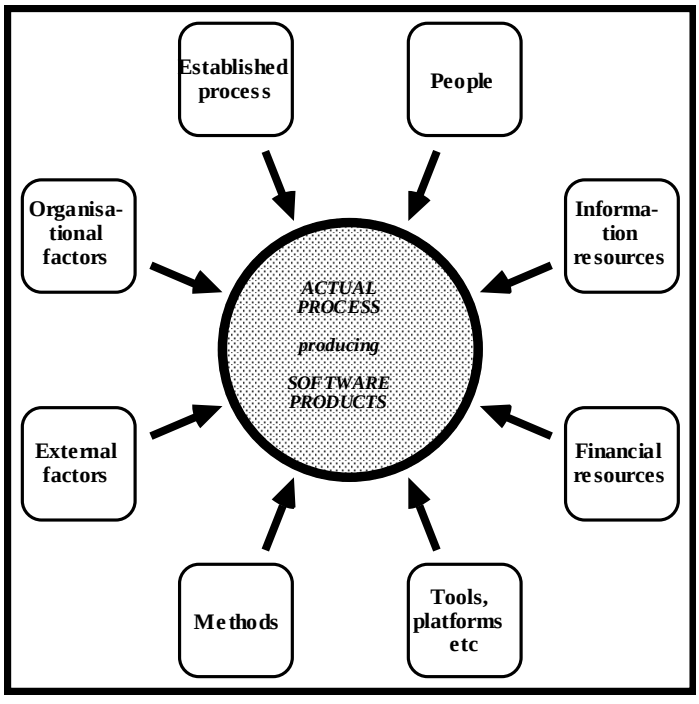


Figure 4 The extended set of factors of production for software

Let us now take stock. Starting from the "pure process" view, as represented by [Paulk 1995], and proceeding via the fairly widely accepted "three factors" view, we have suddenly taken a rather large jump — though not one that will discomfort those familiar with models of quality in other domains.

But we are not finished. Another large jump is needed to complete our voyage. This final jump is necessary to meet three further requirements of a realistic systems model of software production. Those requirements are as follows.

- Because of the way it is drawn, figure 4 cannot account for the systemic relationships between the eight factors of production (in the way that can be done easily when there are only three).
- So far we have accounted for the *production factors* that are the *causes* of the actual process and, as a direct consequence, the software product. But what happens on the far side, as it were: what are the *effects* of the actual process and the product which it generates? Those effects may be called the *performance measures* of the actual process.
- Finally, in order to determine where improvement effort should be targeted among the factors of production, it is necessary to understand how changes in the production factors bring about changes in the performance measures, so as to achieve (to the greatest degree possible) measurement–based and feedback–controlled improvement.

Figure 5 incorporates the appropriate changes to meet those three requirements.

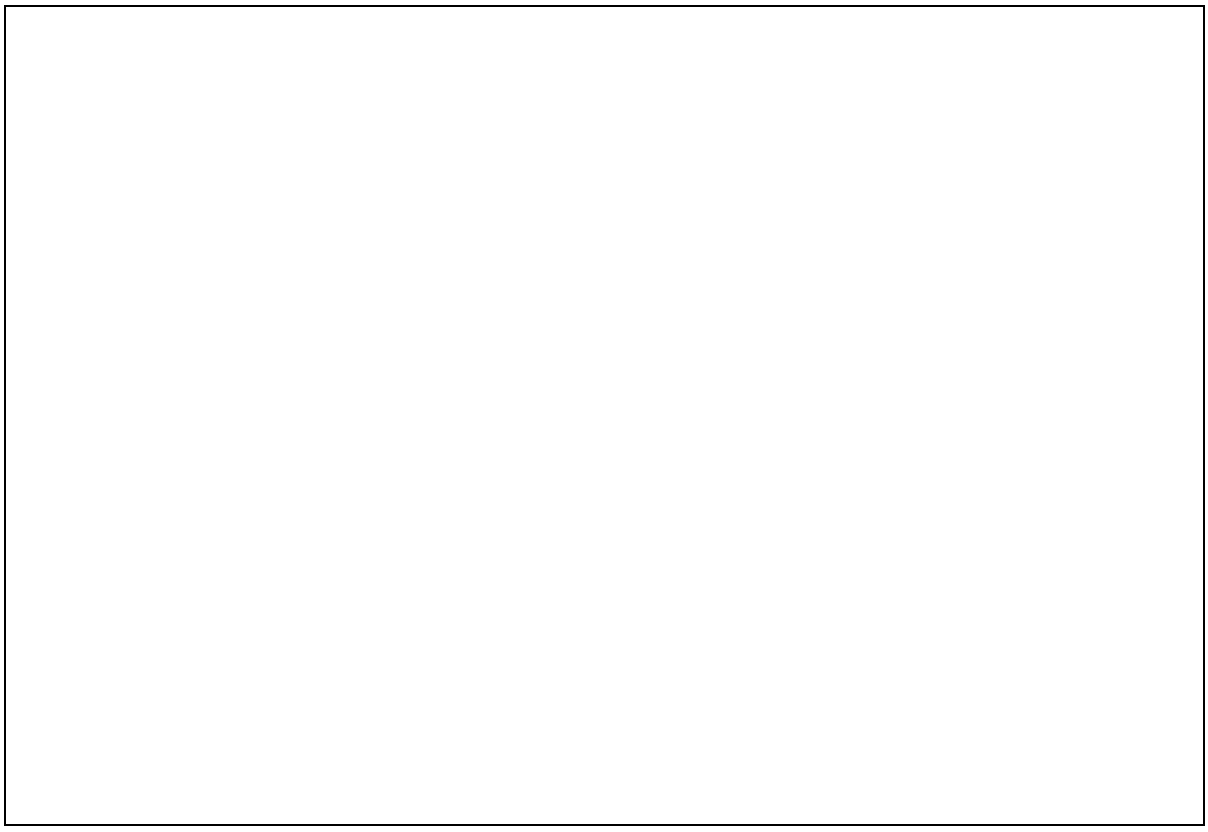


Figure 5 The final model of production factors and performance measures

Throughout the argument up to this point, a continuing emphasis has been put on identifying the total system within which the actual process operates, and on identifying the systemic relationships between its main components. We now have to recognise that the eight production factors and seven performance measures in figure 5 constitute only a first–level decomposition of the total system: they are only the highest–level components.

In the case of the box labelled “established process”, we are accustomed to a far more detailed decomposition (provided by all the process maturity models), through process categories, to process areas and ultimately to base practices. A similar kind of decomposition can be provided for other (though not all) boxes.

For instance, among the production factors a first–level decomposition of "organisational factors" might be as follows.

- Leadership (including culture, management style).
- Internal communication.
- Strategy (including goals, vision, policy).
- Innovativeness (including change management).
- Decision making (decision quality, implementation).
- Organisational structure.

Similarly, among the performance measures a first–level decomposition of product quality, following [ISO 1991], would be as follows.

- Functionality.
- Reliability.
- Usability.
- Efficiency.
- Maintainability.
- Portability.

In seeking to understand systemic relationships in the total system, it is necessary to deal with these decompositions of the main components. Understanding the interactions among the factors of production, for instance, means understanding the relationships among their lower–level components.

We are now beginning to envisage modelling of a considerable degree of complexity, which it is beyond the scope of this paper to investigate in any detail. Suffice it to suggest one simple type of tool, the two–dimensional matrix, which could be used to study relationships between the components of a selected pair of factors. We will show two simple examples.

One relationship between "established process" and "methods" is the "uses" relationship: a particular base practice, for example, uses one or more specific methods (which may be alternatives or may be used together). That relationship could be represented in a matrix of the following form.

	Method A	Method B	Method C				
Base practice 1	x						
Base practice 2		x	x				
Base practice 3				x			
		x					
					x		

Figure 6 Matrix showing "uses" relationship between practices and methods

A different kind of relationship, which might be called "boosters/inhibitors to change" might exist between "established process" and "organisational factors", where specific organisational subfactors might

boost (positive) or inhibit (negative) attempts to improve specific base practices. Boosters might be things such as “clear corporate strategy communicated to software practitioners”. Inhibitors might be things such as “corporate fear culture”. That relationship could be represented in a matrix of the following form.

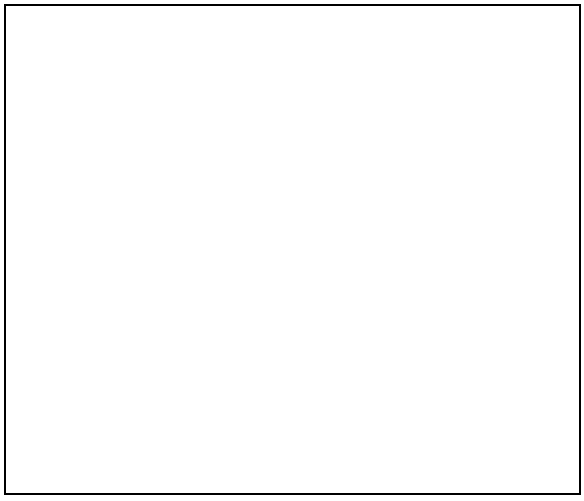


Figure 7 Matrix showing “boosters/inhibitors to change” relationship

Such matrices are just one example of simple modelling tools which could be used to investigate relationships of various kinds operating within the total system modelled in figure 5. Other tools would include, for instance, cause–effect diagrams of the kind shown in [Weinberg 1992].

A backward look at process maturity models

From the systems perspective given by the discussion in the last two sections, let us take a concluding look back at process maturity models as they stand at present. A number of points can be made.

- Process maturity models measure one attribute (maturity) of one factor of production (established process). They take no account of other possible process attributes (such as fitness). By emphasising process, they de–emphasise the other production factors.
- This narrow focus has both a strength and a weakness. The strength is the strength that lies in simplicity: they are simple models which, because they abstract massively from reality, are easily understood. The weakness comes precisely from that massive simplification, if their use is not accompanied by a balanced understanding of how process interacts with other production factors to impact overall performance measures.
- Process maturity models measure maturity by counting the presence or absence of standard practices. This is a very simple means of measurement. One interesting feature is that, by its very nature, it is unable to measure the maturity of individual practices: it can only measure maturity at higher levels of granularity. It also cannot fairly cope with organisations whose set of practices, for good and deliberate reasons, varies from the standard set.
- Process maturity models are unconcerned with cause–effect relationships. They are based on a very simple proposition, that software product quality is a function of software process quality. Such a statement of a static equality can be vastly misleading, because it ignores the real–world chains of events which are interposed between a process change and a product change, the speed at which they unfold, and the extent to which they are affected by other causes.

  To return for a moment to economics, the errors of monetarism as championed by Reagan and Thatcher derived from just such an over–simplification. The simple formula was that the price level is a function of money supply (derived from an ancient formula in economics, which says that the price level times the quantity of goods sold equals the money supply times the rate of circulation).

> This formula has exactly the same shortcomings as the formula for software product quality: it ignores real–world chains of events, the speed at which they unfold, and the extent to which they are affected by other causes. The pursuit of monetarism had some successes, but they derived from other causes than the simplistic theoretical formula on which they were based. The same applies in the case of software process.

Readers might conclude that this paper amounts to an "attack" on process maturity models. That would be a complete misinterpretation. During the present decade, process maturity models, pre–eminently the SEI CMM for Software, have achieved a remarkable uptake, and the available evidence indicates the extent of the improvements which they can assist in achieving. They have also achieved a revolution in management's awareness of, and attitudes to, the potential for bringing the software function under effective control.

This paper's goal is, in recognising those substantial and essential achievements, to look ahead at the requirements and opportunities for building on these achievements in the future. The conclusion it offers is that there is long way still to go in developing methods for understanding the software process, its various measures, its systemic interactions with other factors of production, and their effects on the measurable performance of the software function and the total business.

References

[ISO 1991] **International Organisation for Standardisation.** ISO 9126, Information technology — Software evaluation: Quality characteristics and guidelines for their use. ISO. 1991.

[Paulk 1995] **Paulk (Mark C), Charles V Weber, Bill Curtis, Mary Beth Chrissis**. The capability maturity model: guidelines for improving the software process. Addison–Wesley. 1995.

[Reimann 1993] **Reimann (Curt W).** The Malcolm Baldrige National Quality Award and ISO 9000 registration: understanding their many important differences. Standardization News, November 1993.

[Waldner 1995] **Waldner (Günther).** Total quality management and the European Quality Award. In *Practical improvement of software processes and products: Proceedings of the ESI–ISCN 95 Conference on Measurement and Training Based Process Improvement, Vienna, September 1995.*

[Weinberg 1992] **Weinberg (Gerald M).** Quality software management. Volume 1: Systems thinking. Dorset House. 1992.

BICO **B**enchmarking and **I**SO 9001 CO**mbined**

Hans Scherzer
APAC GesmbH
Mariahilferstraße 89a/44
A-1060 Vienna
apacshz@ping.at

Susanne Lanzerstorfer
QUEST
Unterstinkenbrunn 78
A-2154 Unterstinkenbrunn
101332.2376@compuserve.com

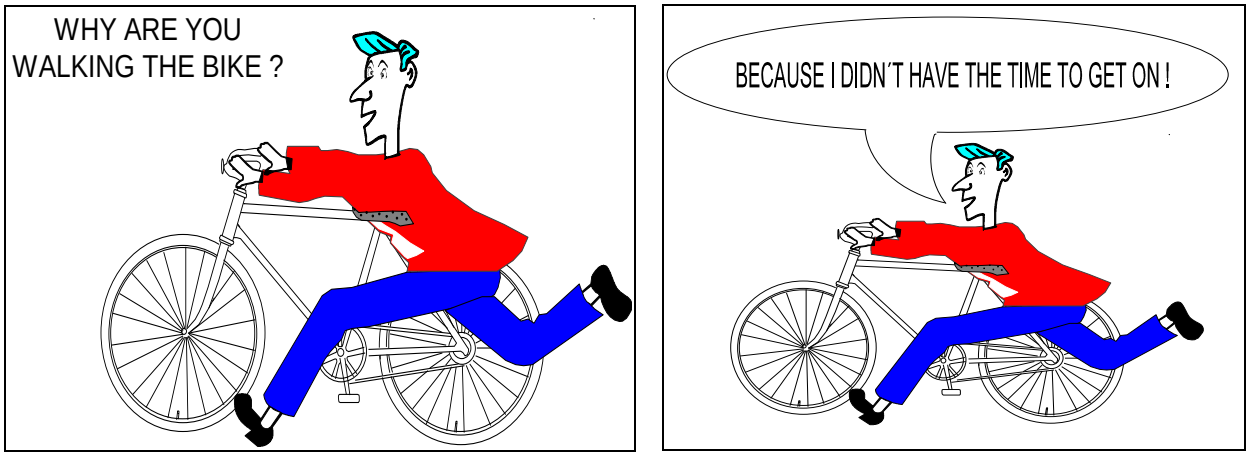
Abstract

BICO (**B**enchmarking and **I**SO 9001 **CO**mbined) is a tool's supported, quantitative method for assessments and process improvement which is in conformance with ISO 9000 ff. and SPICE. Up to now the methods and tools are applicable in environments in which development plays an important role. It is based on an ISO 9001 conformance questionnaire, a tool set that allows fast compilation and visualisation of the collected data, and a formal approach to identify improvement potentials, implement improvement measures and measure success.

Introduction

Background

In many organisation where APAC starts its consultancy activity the situation is similar to the picture shown below:



Some other existing methods that help to get on

ISO 9001
The acknowledged industry standard ISO 9001 doesn't include the quantitative evaluation and only has the test character with the limited option of passing or failing. Due to the lack of numerical evaluation, ISO 9001 audits are only suitable to improve a process up to a certain point, since the extent of the improvement cannot be measured. The benefit is an internationally acknowledged certificate.

SEI/CMM
CMM is a model to measure the maturity level of a software development organisation based on the processes used by the organisation. This model assumes a five level maturity scale from initial (Level 1) to optimising (Level 5) and provides a mechanism for projects which identifies their weaknesses, their maturity level, and required steps for their improvement.

SPICE

Spice, providing a framework for assessments of software processes, will become an ISO-standard in the near future. Being equipped to map the results of an assessment on a six level maturity scale, Spice is also able to identify the improvement potential. Based on a best practice approach the current status of part or of a whole company is determinable from any Spice-conform assessment.

BOOTSTRAP
Bootstrap is based on the CMM/SEI model and offers the numerical evaluation of the quality level of the process (split in system evaluation and project evaluation) similar to the ISO 9001 arrangement (attributes are about equal to the norm-elements). It is possible to get profiles of the process and the project. A Bootstrap-Assessment is a good basis for improvement, that is able to be measured.

Why BICO

The idea was to develop a method to support small companies on their way towards process improvement. At the time when BICO was developed SEI [09] was on the market and the BOOTSTRAP project [11] was just finished. Some other models were also available (e.g. Trillium [06]).

The most important reasons for developing BICO were the following:

1. Using SEI or BOOTSTRAP for process improvement is often too expensive for very small organisations. Many of organisations are quite small in Austria (1-5 employees).
2. SEI and BOOTSTRAP are oriented towards software development and therefor can not be applied to companies which develop systems consisting of software and hardware. Most of APAC's customers develop software and hardware. It is important for APAC's customers to maintain and improve a quality system which covers the entire system and not only software.
3. Neither SEI nor BOOTSTRAP are well known in Austria. For most companies it is more important to become ISO 9001 certified, than to proof a certain maturity level.

BICO overview

BICO is a tool's supported, quantitative method for assessments and process improvement which is in conformance with ISO 9000 ff. and SPICE. Up to now the methods and tools are applicable in environments in which development plays an important role. The method is based on an acknowledged quantitative assessment model and resulted in an ISO 9001 conform questionnaire. A Toolset (ASAP) that allows fast compilation and easy to understand visualisation of the data collected during the audit supports the process improvement. The quantitative assessment results tell the customer where he is and identifies his improvement potential. The BICO - model guides him on the way towards higher quality and productivity by process improvement based on the assessment results and the business goals.

The benefits are:
- The audit process' results are strength/weakness profiles of the system and of the evaluated projects.
- The profiles visualise clearly where the ISO 9001 requirements are satisfied and give the customer enough information to start with process improvement.
- As we map the results of our method onto the SEI Capability Maturity Model we don't stop at ISO 9001 level, but we support the customer on his way to TQM.
- A well-defined method helps you to identify the most effective improvement measures. BICO helps you to increase quality AND productivity.
- The benchmarking tells you where you are, absolutely and in comparison with previous audits of the same unit and other projects.
- BICO benchmarking includes process, product and commercial figures.

BICO Life Cycle

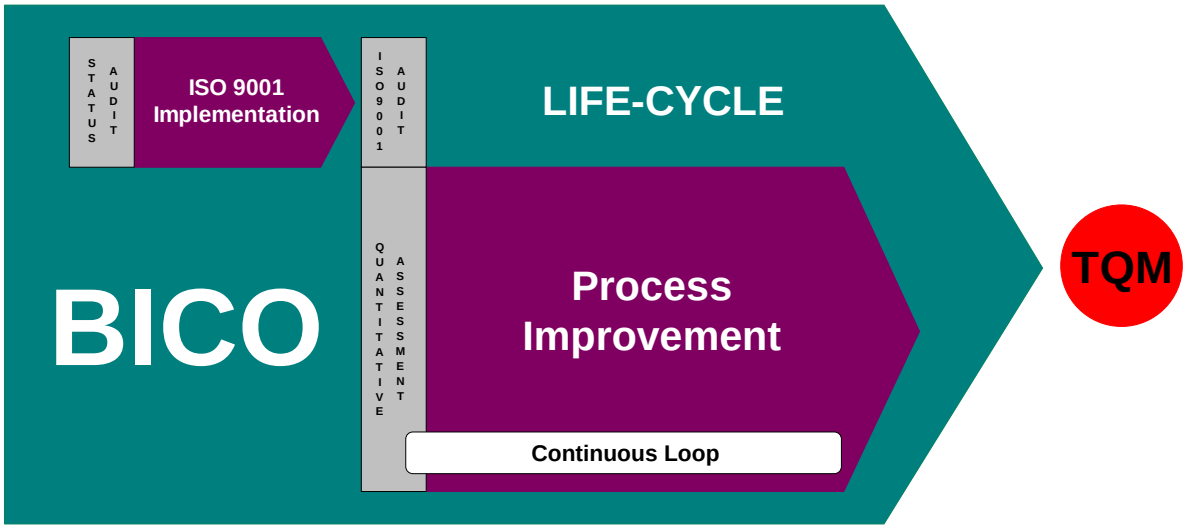
Overview

BICO is a modularised approach that allows perfect customising to the customers needs. BICO is not only a method and a toolset for quantitative process assessments but

supports continuous process improvement. It is based on the ***plan - do - check - act*** approach. The major goal of the BICO process is to improve quality **AND** productivity. The BICO material is structured into the following modules:

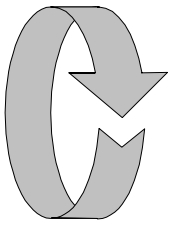
Phase Independent:		**Phase Dependent:**	
BLCM - PM	**Project Management Module**	**BLCM - PIP**	**Process Improvement Preparation; Planning and Design of the Improvement Process**
BLCM - PC	**Project Controlling Module**		
BLCM - QA	**Quality Assurance Module (includes overall V&V, BLCM metrics and BLCM V&V)**	**BLCM - PI**	**Process Improvement Module**
BLCM - DB	**Database Module (data stored without reference to the source)**	**BLCM - A**	**Audit Module (includes the tool ASAP-F and ISO 9001 conform questionnaire)**
BLCM - PRO	**Product Management Module**		
BLCM - TE	**Training and Education Module**		
BLCM - IF	**Interface Module (providing interfaces to map the results on other quality assurance system models)**	**BLCM - AN**	**Analysis Module (includes the evaluation tool ASAP-B)**

The BICO Life-Cycle which is outlined below is driven by business goals.



The BICO Life-Cycle Improvement Process is based on:

- a well defined process
- business goals compiled into priorities
- extensive validation and verification
- customers feedback
- conformance with ISO 9001



Plan
- **Process Improvement Preparation, Planning and Improvement Process Design**

Do
- **Process Improvement**

Check
- **BICO Audit (includes planning and data compilation) - results in certificates**

Act
- **Audit Results Analysis**

BICO Audit

The BICO audit process, which is well defined and documented in the module BLCM - A, is supported by a tool that runs on a small palmtop PC for on-site data collection. The tool accepts questionnaires in several formats without any restrictions on the contents or

structure. APAC uses a proprietary questionnaire that takes into account ISO 9001, ISO 9000-3 and CMM/SEI. Two different questionnaires are used for project assessments and system assessments. Both questionnaires consist of about 200 questions. The compilation and visualisation of the results is performed with another tool that runs under Windows '95 and offers state-of-the-art simulation and visualisation. The compilation is based on a proprietary algorithm that allows scoring in higher levels, even if the requirements of lower levels are not fully satisfied. The output may be structured in several ways, and the most commonly used structure is according to ISO 9001, attributes. The resulting profiles will be discussed in the case study. The scoring ranges from 1 to 5 with a granularity of 0.25.

The module BLCM-A supports the following types of audits: quality system audit (entire organisational unit), project quality system audit (project), project status audit (plan versus actual of schedule, costs and resources), evaluation of the process capability of proposers (at large international competitive invitations to tender) and any combination of the types mentioned above.

The process was designed to meet the requirements of very small or small organisations that cannot afford a full-scale SEI-assessment or similar expensive approaches. The main purposes of most of the BICO audits are to identify weaknesses and improvement potentials, to visualise the results in a way everybody understands and to buy-in for the process improvement process. The audit process meets these requirements and sacrifices high accuracy for suitability and applicability. The process also meets the requirement to assess the entire organisation and not only the software producing unit. Most of the assessed organisations design and produce software as well as hardware. The disadvantage of this approach is that the results are not comparable with the results of other assessment methods without additional data selection and compilation. The benefit of this approach is that the customer's needs are met. The BICO method being based on the experience of the authors shows that process improvement in small organisations can be achieved more efficiently and effectively, if the entire organisation is involved. The BICO audit process also resembles the experience of the authors, namely, in that strict accuracy doesn't pay in terms of the necessary effort required to achieve it. It is much more important to trigger improvement, to buy-in and to identify large improvement potentials. If one visualises the progress with a profile one year later, this is more help in increasing the motivation to support the improvement process than strict accuracy.

Under normal conditions the BICO audit method allows assessment of an organisation of about 20 employees within the following time constraints: planning of the audit, evaluation of the quality management documentation 8 hours, on-site assessment 8 hours, data compilation and report generation including a proposal for process improvement actions 8 hours, presentation and discussion of the results 4 hours.

The ASAP-B Tool (Visualisation of the results)

The name ASAP-B is derived from: **A**udit **S**upport & **A**nalysis **P**rogram - **B**ackend. This tool is the successor of the MsExcel - based prototype known as EST. Like EST, this tool provides histograms representing the characteristic profile of the processes of a company but it also offers additional features.

The context in which ASAP-B is used

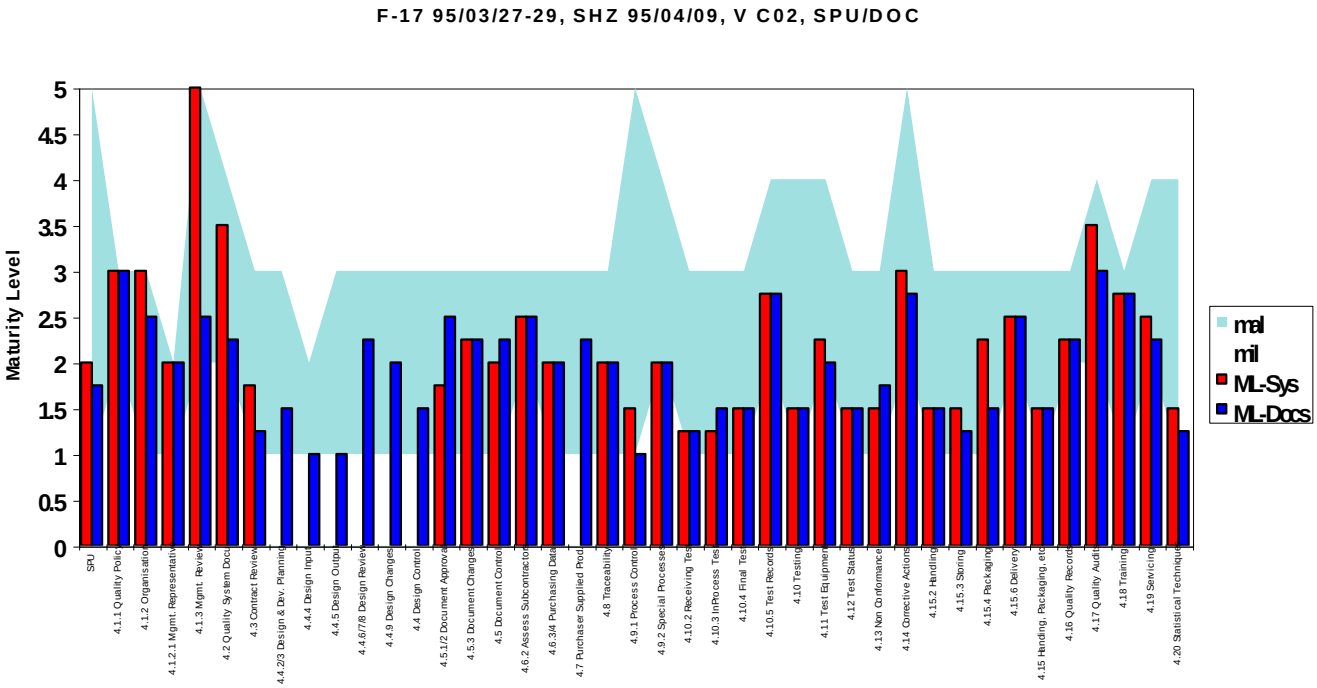
The audit process is supported by ASAP-F (**A**udit **S**upport & **A**nalysis **P**rogram -**F**rontend). Before the auditing begins, the organisation audited, the questionnaire used, the auditors, the place and time and some additional information about the organisation audited have to be determined and stored. The questionnaire to be used and the definition of the subset of questions of the questionnaire are an input information for ASAP-F. During the audit the questions are

displayed by ASAP-F. For each question the evaluation result and remarks, if appropriate, are entered into ASAP-F. These results are stored in a result file which is the input file for ASAP-B. The information about the organisation audited and the information about the audit itself are also processed by ASAP-B.

Output of ASAP-B

Based on the input information provided by the result file of ASAP-F and the questionnaire, ASAP-F calculates the profile. Using the result file ASAP-B calculates the results according to the definitions in the questionnaires and parameters defined by the user. ASAP-B visualises the results. The information about the audited organisation and the information about the audit itself are also visualised.

The ASAP-B offers several options for structuring and displaying the results. The tool also offers the possibility of simulations. Following an example that shows an audit result structured in accordance with ISO 9001.



Explanation of the major elements of the chart, visualising the scores of the quality management system documentation and the quality management system implementation:

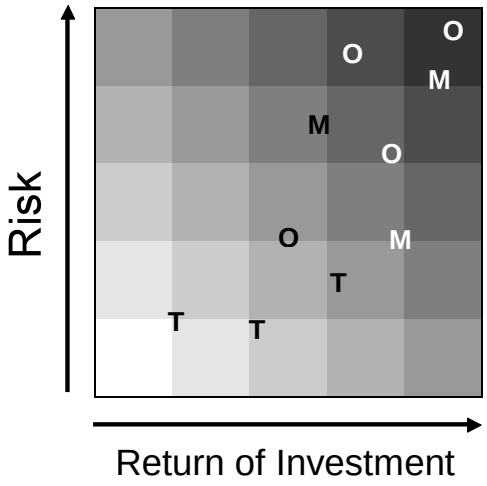
- the light grey area shows the possible scores depending on the attribute
- the middle grey bars show the values scored by the quality management system documentation
- the dark grey bars show the scores of the implementation of the quality management system
- on the bottom of the chart the ISO 9001 quality attribute, to which the bars belong, is shown

No scores of the quality management system are available for the attributes design and purchaser supplied products, as these attributes are calculated and visualised at project level.

BICO Improvement Process

The BICO improvement process is based on the rule **O > M > T** that means that organisational aspects are more important than methodology, and methodology is more important than technology.

Much attention is paid to the process of the selection of the improvement measures and process improvement planning.



Selection of the improvement measures

Taking into account the rule that for any process

- **the organisation is more important than the method**
- **the method is more important than the technology**

The prioritising of the improvement measures is based on:

- **the audit results**
- **the business goals**
- **the risk involved, if the measure is not implemented**
- **the return of investment gained by the implementation of the improvement measure**

By the procedure for selecting the improvement measures as outlined above the customer and the consultant compile the audit data into the appropriate improvement measures.

Part of this process is also the establishment of verifiable goals for the improvement measures to determine success and failure of the measures. The goals for the improvement measures are always based on the business goals and not only on a maturity level which is targeted.

BICO Project Audit

Mostly the goal is to get figures and facts to minimise the project risk. The project audits aim at low cost, high efficiency and reliable figures. BICO supports three kind of project audits.

Project Progress Audit: A Project Progress Audit covers schedules, costs, resources and progress. The plausibility of the planning is checked, the actual is compared with the plan and the result is described in a detailed audit report. The audit report provides you with the following information on task, workpackage and project level:

- plausibility of the plan
- plan versus actual of schedule, costs and resources
- estimated time to complete
- estimated effort to complete

The audit report contains neither a maturity profile nor information on compliance with quality standards.

Project Quality Management Audit: A project is audited and benchmarked against a predefined model and scale (e.g. SEI/CMM). Additionally the conformity with applicable standards (e.g. Quality Manual, ISO 9001, MIL-STD-498, project specific standards) is determined. The output of the audit is an audit report that includes:

- a maturity profile for the actual process
- identification of strengths and weaknesses
- identification of the major project risks
- proposed project specific improvement measures that are necessary to meet the project goals

Project Status Audit: This is the combination of the Project Progress Audit and the Project Quality Management Audit that provides a complete description of the actual project status.

Experience with BICO

BICO Audits

The following BICO audits have been performed since the start of the BICO project:

QMSA	quality system audit	10
PMSA	project quality system audit	9
PSTA	project status audit	4
PRPA	evaluation of the process capability of proposers	2
MIXA	any combination of the types mentioned above	1
	TOTAL	26

The audits were split among 9 organisations. The following table provides some information about the target organisations:

Code	Employees	ISO 9001	Area	Audits	Trigger
A	20	no	image processing	QMSA(1)	customers, ISO 9001
B	7	yes	quality engineering	QMSA(2), PMSA(1)	process improvement
C	12	no	banking devices	QMSA(1), PMSA(1)	customers
D	100	yes	lottery systems	QMSA(2), PMSA(1)	process improvement
E	6	no	embedded systems	QMSA(2)	process improvement, ISO 9001
F	150	yes	air traffic control	QMSA(2), PMSA(6), PSTA(2)	process improvement
G	500	no	air traffic control	PRPA(1)	risk minimisation
H	100	no	air traffic control	PRPA(1)	risk minimisation
I	20	no	air traffic control	PSTA(2), MIX(1)	customer

Process improvement based on BICO audits (BICO-Life-Cycle)

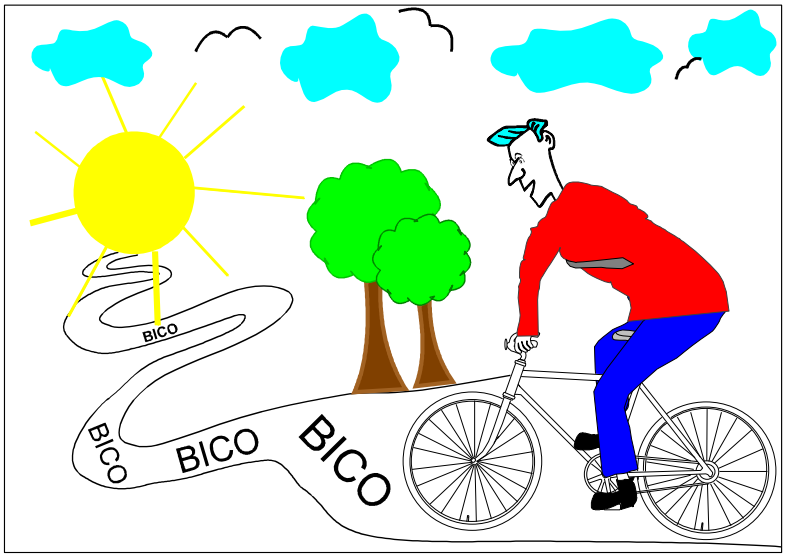
APAC and QUEST have performed ten quality system audits until now. Five audits were followed by process improvement actions. Three audits were a follow on action after process improvement actions. At four organisations APAC or QUEST supported the improvement process. All improvement processes that were supported by APAC or QUEST were successful, which means quality AND productivity were improved.

Future Outlook

It was already identified during the field test that the training module was too weak. Many customers asked for training that couldn't be provided. This was the reason for APAC and QUEST to participate in the PICO project. The PICO project is partly funded by the LEONARDO program (EC) and will develop a comprehensive and consistent set of training material that covers the entire process improvement cycle including the audit process.
It was also identified that the BICO method isn't limited to software development organisations, but that it is only applicable in organisations in which development plays an important role. A proposal was submitted to the Austrian authority for funding of an extension of BICO, to cover also less developmental-oriented environments. Of course, this extension cannot be based on the SEI/CMM model, and an new model has to be defined. We expect the first results within the next year. This is a very interesting extension, as no quantitative assessment model besides developmental-oriented environments could be found.

Conclusion

BICO is a very effective and efficient way to get on



References

[01] Design of a Quantitative Quality Evaluation System (QUES), Dissertation of R. Messnarz, December 1994.

[02] EN ISO 9001:1994

[03] ISO 9000-3 (Part 3, Guidelines for the application of 9001 to the development, supply and maintenance of software)

[04] ESA PSS-05-0, Issue 2, Software Engineering Standards

[05] IEEE Software, July 1994, Bootstrap: Fine-tuning Process Assessment, V. Haase, R. Messnarz, G. Koch, H.J. Kugler, P. Decrins

[06] Trillium Questionnaire, Telecom Software Product Development Capability Assessment Model, Version 3.0, 1994

[07] DIN ISO 10011 Guidelines for auditing quality systems, Part 1-3:1992

[08] EN ISO 9004-1Quality management and quality system elements - Part 1: Guidelines :1994

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

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

[11] Software Process Assessment and Improvement / The Bootstrap Approach, Pasi Kuvaja a.o., ISBN 0-631-19663-3, 1994

[12] TickIT, Guide to Software Quality Management System Construction and Certification using EN 29001, 1992, dti

[13] Positive correlation between process maturity and product quality as well as project performance, 1996, S. Lanzerstorfer

The ESSI Process Improvement Experiment „ReUse“

Elisabeth Kauba
Siemens AG Österreich
Program and System Engineering Division
Gudrunstrasse 11, A-1100 Vienna
elisabeth.kauba@siemens.co.at

Abstract

This paper presents the results of an ESSI process improvement experiment (PIE), analysing the potentials of reuse and how to employ reuse strategies.

Reuse is one of the most promising ways to improve productivity and quality, and it is the one technology with the highest return-on-investment: According to Capers Jones of SPR, the successful enactment of a full software reusability program can return 30,-$ for every 1,-$ invested. On the other hand, the implementation of systematic reuse is risky, and costs are uncertain. So, the actual cost/benefit ratio is difficult to assess, and consequentially, management is reluctant to commit themselves to the initiation of cultural change towards a reuse-centric software engineering process.

The project ReUse shows that the establishment of a reuse-centric software engineering process, in other words the improvement steps from ad-hoc reuse to a managed reuse process, requires a carefully planned and executed strategy for the introduction of reuse into an existing software producing organisation and process. The introduction of systematic reuse requires change management with respect to project management and responsibilities, organisation, process model, metrics, people skills and motivation, and the adoption of new methods and tools.

The ESSI Process Improvement Experiment „ReUse“:

During the first part of the project, the project organisation and the Siemens - PSE software engineering method SEM have been temporarily enhanced with reuse activities, managment functions and metrics. After training and motivation of all hierarchical levels involved, an initial set of reusable assets (software and other results of the software engineering process) has been submitted to the new central reuse library - the assets have been analysed, classified, described, and archived.

The second phase of the experimentation comprises development with reuse (both projects use assets from the reuse library, emerging from the own and the other baseline project), and development for reuse (both project teams bring in the results of their ongoing baseline projects, and thus further enlarge the common reuse library with assets meeting the requirement of improved reusability).

The ESSI process improvement experiment (PIE) „ReUse“ was carried out on the baseline of one business oriented (planning system) and one technically oriented project (computer integrated telephony), but the results can and will be replicated for embedded & control systems, as well.

Exploitation of Lessons Learnt:

Siemens - PSE has gained experience with the introduction of reuse in a software producing organisation, and learned about reuse barriers and how to overcome them.

The results will be used to improve productivity and quality throughout the own organisation by establishing a refined permanent reuse organisation, reuse method and reuse process. Further to that, the experiences and the acquired expertise in establishing a managed reuse process will be offered to software producing organisations throughout the international Siemens group and to the wider european community.

The project is funded as Process Improvement Experiment (PIE) by the Commission of the European Community (CEC) under the ESSI programme: European Systems and Software Initiative. The goal of the ESSI programme is to promote improvements in the software development industry so as to achieve greater efficiency, higher quality, and greater economy.

Introduction

Siemens PSE

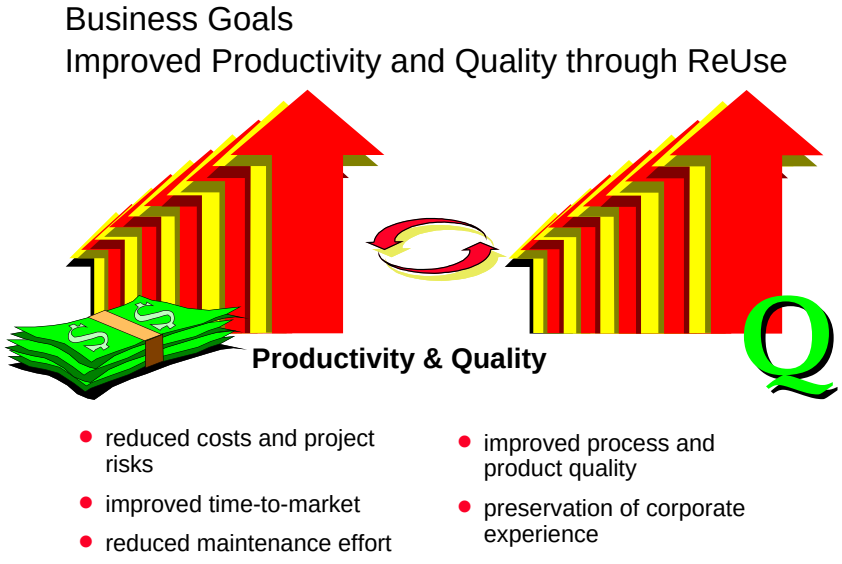
Siemens AG Österreich is the Austrian daughter company of the international Siemens Group, which is with 373.000 employees one of the leading providers of electrical engineering and electronics. Software and especially an efficient software engineering process give Siemens the competitive edge to succeed in innovative and challenging markets like energy, telecommunication, industrial automation, medecine, traffic and information technology.
The Program and System Engineering Division - short PSE - of Siemens AG Österreich is with 3.500 software engineers the largest software and IT services provider for the international Siemens Group and Siemens customers around the world. Software plays an important role for most of the products Siemens is delivering to the world market.
The main business of Siemens PSE is the development of software embedded in Siemens products (telecom, process automation, medical equipment, and many more), as well as the development of customized solutions for Siemens and Siemens customers in practically every line of industry (manufacturing, banking and insurance, service sector, authorities,...).

Siemens PSE is committed to quality and continuous quality improvement. Since October 1993 Siemens PSE holds the EQNET ISO 9001 (EN 29 001) certificate registered with number ÖQS-Reg.No.: 106/0.

Business Motivation

A series of assessments based on the Carnegie Mellon "Capability Maturity Model" confirmed that Siemens - PSE has already reached a top position in software engineering compared to international software developing organisations.

Nevertheless, since besides the established competition in the homemarket, US and Japan, new competitors emerge from the youngest "IT-nations" like India, S-Korea, Taiwan, ... it is definitely necessary to further improve the already high level of quality and productivity, thus to be able to deliver to the customer better quality in shorter time and for less money than the competitors can.



To further improve the Siemens PSE software engineering process, a catalogue of recommendations, resulting from the CMM-assessments, was established and is, after

successful evaluation, being implemented step-by-step. Among the most important recommendations, Siemens PSE is committed to follow, are the following:

- installation of a central reuse management
- introduction of domain-specific software reuse managers
- evaluation and (if positive) introduction of inter-departmental, inter-domain reuse supported by a "Reusable Assets Broker"
- design and implementation of a detailed reuse model
- application of reuse metrics
- assure the economic efficiency of reuse

Reuse Goals

The expected outcomes of the experiment were

- the validation respectively refinement of an existing theoretical concept for the introduction of systematic reuse into an existing software engineering process
- related workproducts
  - SEM-R, a reuse oriented software engineering process model
  - a reuse organisation model
  - reuse training and motivation programs
  - guidelines for design, development and documentation for reuse
  - a reuse infrastructure consisting of a central reuse library, methods and related tools supporting the forward and reverse/re-engineering of reusable assets
  - reuse metrics (including procedures for data collection and reuse control)
- experiences with
  - the management issue (reuse barriers and how to overcome them):
  - the technical impact (benefits, preconditions, consequences,...)
  - the commercial impact (up-front investment, risks, amortisation time, benefits,...)
- the improvement of reuse maturity from ad-hoc reuse to systematic reuse within the baseline projects

The Experiment and the Baseline Projects

The PIE was carried out in three phases: the preparation, execution and evaluation of the experiment.

During **Phase 1 - Preparation of Experiment** - all actions have been taken to assure that all pre-conditions for a successful experimentation are given. These actions comprise organisational measures (set-up of a temporary reuse organisation), process improvements (enhancement of the existing process model with reuse activities), the introduction of new techniques, guidelines, tools and metrics, and cover the human aspects through a comprehensive training and motivation program. The first reuse activities are undertaken by building an initial stock of reusable assets.

During **Phase 2 - Execution of Experiment** - the main task, the systematic, methodical and controlled performance of reuse activities, was performed. This comprises the reuse of assets already contained in the reuse library on the one hand (search, evaluation, retrieval, adaptation/reengineering), and the building of new reusable assets, meeting the requirements recognised during domain analysis, designing, developing and documenting these assets according to the DDDR guidelines and their submission to the reuse library (classification, description). The experiences gained and measured at defined milestones provided the necessary feed-back for continuous refinement of all improvement measures.

During **Phase 3 - Evaluation of Experiment** - the lessons learnt from the experiment have been evaluated and translated to a plan for future process improvement strategies. Reuse will be introduced also in the other business areas of Siemens-PSE and in other divisions of the international Siemens group. Internal and external dissemination actions assure that the wider european community will benefit from the experiences gained.

The ESSI process improvement experiment (PIE) „ReUse“ was carried out on the baseline of one business oriented (planning system) and one technically oriented project (computer integrated telephony), but the results can and will be replicated for embedded & control systems, as well.

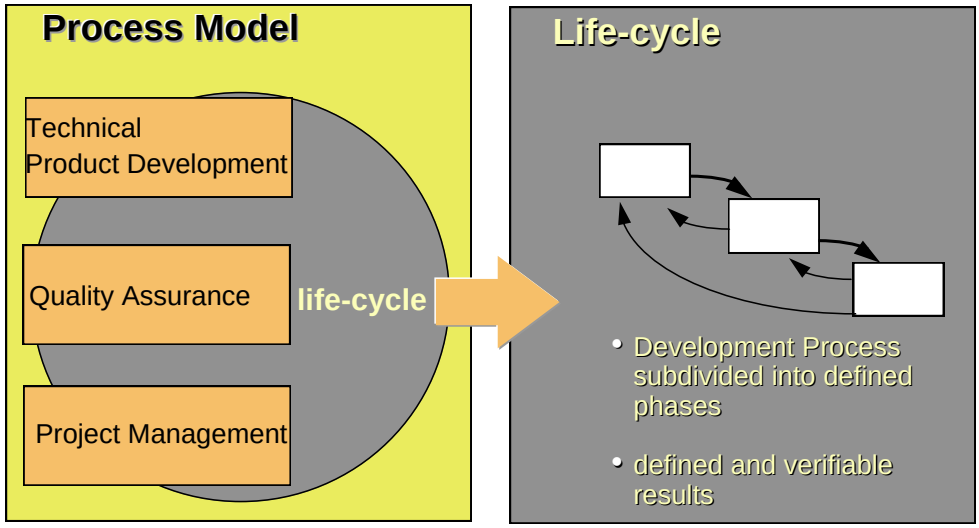
The Process Improvement Experiment

Organisation

Process Model, Life-Cycle

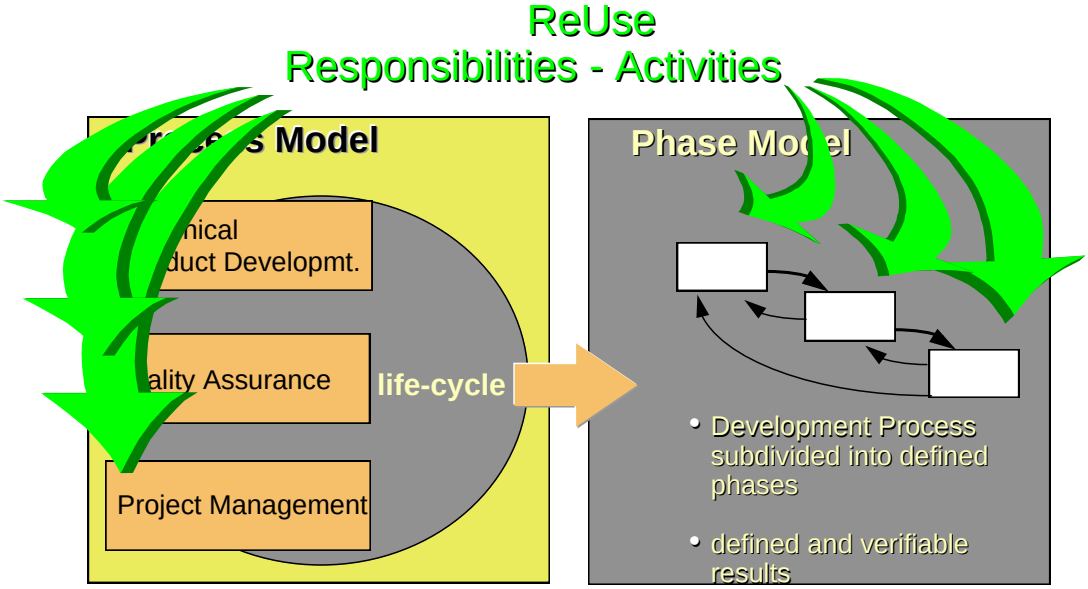
Siemens - PSE has established a proprietory ISO9001 conform process model System Engineering Methodology (SEM) which is mandatory for all Siemens - PSE projects, where the customer does not require the application of his own method, e.g. ESA - PSS05. SEM divides the system development process into a number of major phases, each consisting of several steps. For each of this steps the preconditions are specified, the phase specific actions are defined and the required results are identified. SEM is a waterfall model, specific reuse activities in the different phases of the life-cycle are not foreseen.

SEM - System Engineering Method of Siemens - PSE
a defined process model (ISO certified)



To make developers aware of the necessary reuse activities and stimulate them to perform them, it was necessary to enhance SEM with phase-specific reuse activities, building a reuse version of SEM: SEM-R. Additional procedures for the execution of reuse activities, the collection of data for the calculation of reuse metrics, and the assessment of experiences have been established.

SEM-R - SEM for ReUse-oriented SW Engineering



The enhancements of SEM to SEM-R comprise

- enhancements of the process model
  reuse activities including the required preconditions and the defined deliverables have been added to all phases of the process model
- instruments supporting the practical application of SEM respectively SEM-R have been added or adapted. They comprise new templates for a ReUse-Plan and a ReUse-Report, adaptations of the existing templates for project planning, quality assurance planning, new or enhanced checklists for reviews, progress assessments, project experience assessment
- enhancements of project management with respect to project estimation, planning and tracking
  the nomination a reuse manager, the estimation of the reuse/reusability potential, costs, efforts, required time and expected benefits with respect to reuse, planning of reuse activities
- enhancement of quality assurance and project control
  additional responsibilities and activities, collection of additional data for reuse metrics

SEM-R is still a conventional sequential approach, which is successfully applied to projects where very precise and comprehensive domain expertise is given.

It is recognised, that the waterfall model has its general drawbacks, e.g. for projects confronted with changing user requirements or changes in the environment, and weaknesses, that limit reuse and reusability. In addition to that, iterative models tend to produce monolithic system structures with specialized individual components, limiting reusability. Iterative life cycle models, like rapid prototyping and the spiral model [BOEHM88], or the cleanroom approach, which is similar to the spiral model, but restricts the incremental approach to the implementation phase (not the analysis) [MILLS87] have the flexibility required for development with or for reuse.

To keep the focus of this PIE on reuse, the decision was taken, to accept the drawbacks of the conventional sequential approach, which are in fact minor, given the relatively small size of the baseline projects and the ample domain expertise in the respective application domains, and to perform all reuse activities according to
SEM-R.

However, the necessity to adopt more flexible process models, that are better supporting OO projects, changing user requirements and especially development with or for reuse, has been recognised and the plan for future experimentation with and adoption of an iterative life-cycle model has been included into the plan for future improvement actions.

ReUse Organisation

A temporary reuse organisation has been set up for the PIE:

- a central reuse task force has been installed to motivate, train, support and coordinate the two baseline projects carried out by two different departments

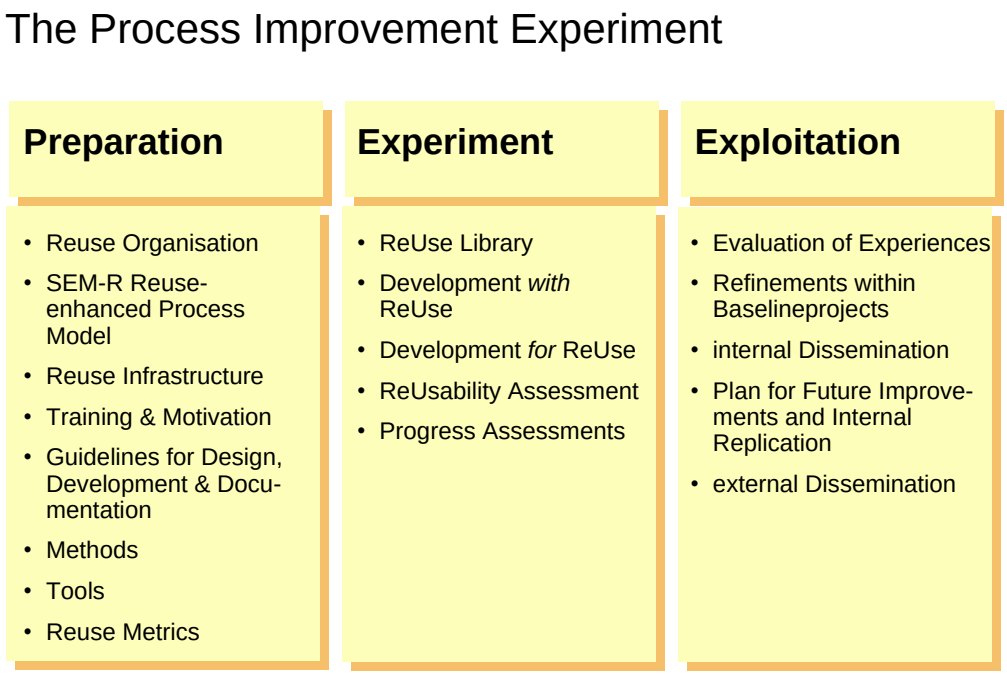
  a special responsibility within the reuse task force is the RAB - reusable asset broker - who acts as catalyst for the reuse process, performs the tasks of the reuse librarian, i.e. the initial set-up of the classification scheme and its continuous refinement, the organisation of the asset archive, notification about new assets, etc., as well as motivates and supports the baseline project team members

- a reuse manager has been named for both of the two baseline projects

Further improvements to the reuse organisation may be achieved through closer integration of reuse responsibilities with CM responsibilities.

Thus, the suggestion for further improvement in the outline plan for future actions will be the establishment of a central reuse manager and a reuse manager for every business area/field and project. The RAB function should be institutionalised, and complemented by a "ReUse Support Centre", covering the growing training and support needs, when more projects respectively business areas will adopt systematic reuse. The responsibility for the introduction of reuse in the different business areas should not be delegated to technical staff, but be directly taken by the responsible business area managers.

Technical Approach



After preparing the experiment by establishing the above mentioned supporting work products, the experimentation with systematic reuse has been started.

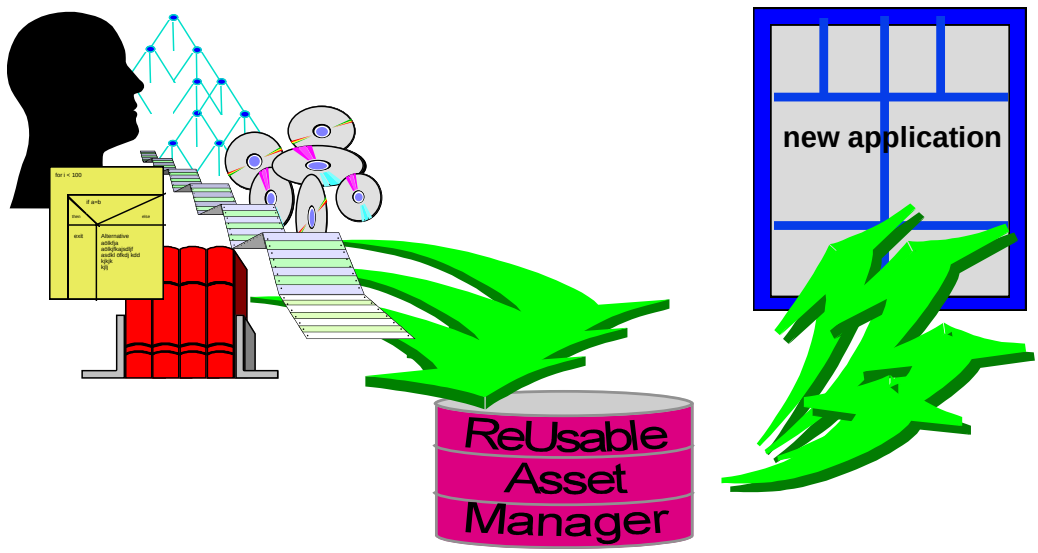
Reusable Assets

The experiment did not restrict itself on code reuse, but took into account all workproducts of the life cycle, including requirements, analysis & design models, code, documents, tests, and the experience and skills of people, as well.

Different reuse approaches are applied:

- development *with* reuse [SINDRE95]
  - reuse library (faceted classification scheme [DIAZ85])
    a repository of reusable assets has been assembled after performing „asset mining“ in predecessor projects of the baseline projects. The identified assets have been described, classified using a faceted classification scheme, combined with free keyword description and full-text retrieval, and archived on a central repository server.

Development with ReUse - Types of ReUsable Assets



- reverse engineering, re-engineering

  Since most of the identified assets have not been developed in the past for multiple use, reverse engineering and re-documentation have been used to facilitate re-engineering (adaptation of the reusable assets according the new requirements) by better code comprehension and documentation.

- reusability assessment

  for every reuse candidate it is necessary to make a technical and economic decision, whether the candidate is worth reusing. Static analysers, automatically calculating metrics about portability, complexity, readability, and many more, as well as checking the conformity to coding standards, have been used to cope with this task.

- white-box reuse

  the reusable assets are not used as they are, but adapted to actual requirements. This asks for code/asset comprehension facilitated by code analysis/reverse engineering tools, making the re-engineering task more efficient

- development *for* reuse [SINDRE95]

- domain analysis

  the two application domains have been analysed to find generalities and variabilities

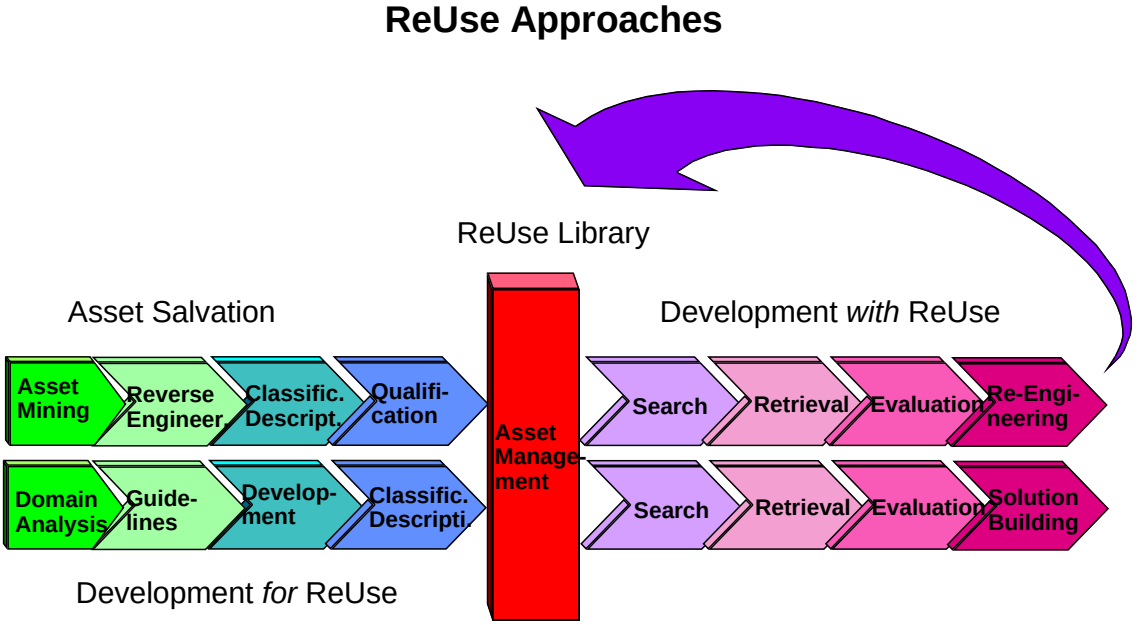
- DDDR guidelines

  one of the baseline projects, using structured technologies, is developing the new workproducts according to guidelines for design, development and documentation, thus assuring improved reusability. This new assets are added to the reuse library.

- frameworks
  the baseline project, using OO technologies, is developing a framework for part of the solution, thus trying to achieve a maximum reuse ratio in future projects.

- grey-box reuse, black-box reuse

the assets developed in the frame of the baseline projects are characterised by improved reusability through adequate design, conformity to coding standards, adequate interfaces and explicitely design adaptation facilities, as well as appropriate documentation. This makes it possible, to reuse asssets as they are, or adapting them only using the provided „hot spots“.



Skills

To enable a professional execution of the experiment, training and motivation was provided on all hierarchical levels, to management and technical personnel. Thus,

- management awareness could be raised
- awareness, motivation and technical skills of the technical staff (software engineers, project managers, configuration managers, quality assurance responsibles and reuse managers) could be enhanced by the training and motivation program.

  Additional skills comprise asset mining, classification and description of reusable assets, search strategies, reverse engineering, re-engineering, re-documentation, asset reusability assessment, application of guidelines, ...

One unexpected result of the PIE was the finding, that the introduction of reuse is an unorthodox, but efficient maturity assessment method, as hidden weaknesses become apparent very quickly. Technically and economically efficient reuse requires at least a medium maturity level. Of course, reuse can be practiced also on lower maturity levels, but the economic benefits will not show in the expected range, when assets are reused, that are not worth reusing. Systematic and economically efficient reuse requires a certain level of maturity and the application of best practices especially in the areas of project management, configuration management, domain analysis, (re-)documentation, reverse engineering and application of standards.

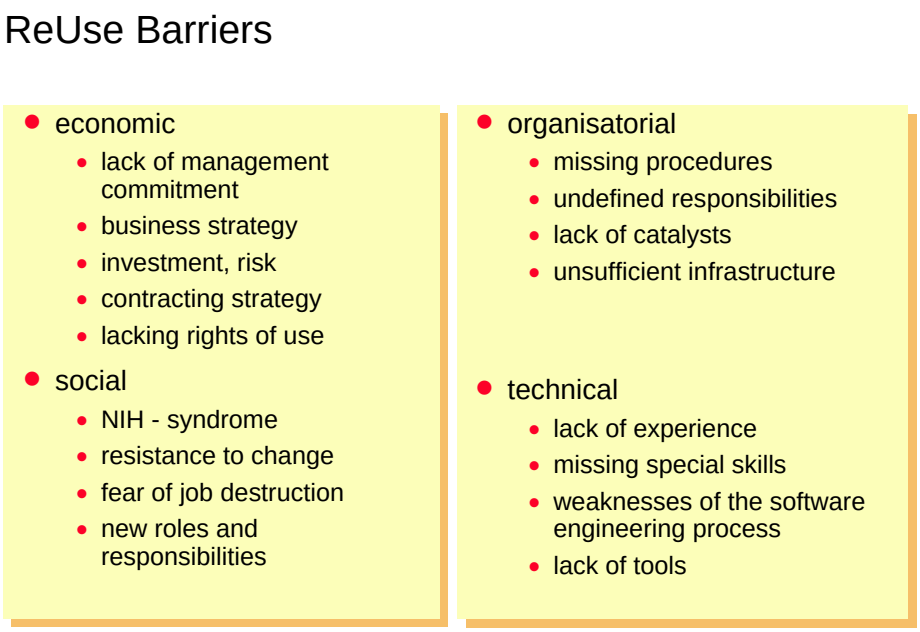
The resistance to a changed way of working had to be met not only by training action, supplying enabling technologies and guidance (process model, methods, tools and

guidelines), but also by motivation and awareness raising actions, especially showing the benefits of systematic reuse.

Culture

One of the major experiences of this PIE was the issue of reuse barriers, and ways to cope with them.

During progress assessments, a long list of potential and actual reuse barriers/ inhibitors was established, ranging from the well known NIH (not invented here) syndrome to fears with respect to personnel reduction ("ReUse is a Job Killer") to less psychological and more managerial aspects as contracting and business strategy to technical/quality related aspects as lack of tools, methods and hidden weaknesses.



Resistance to perform the required systematic reuse activities could be observed especially in the smaller baseline project, where all team members dispose of direct personal communication means, and therefore did not recognise the necessity of organised reuse.

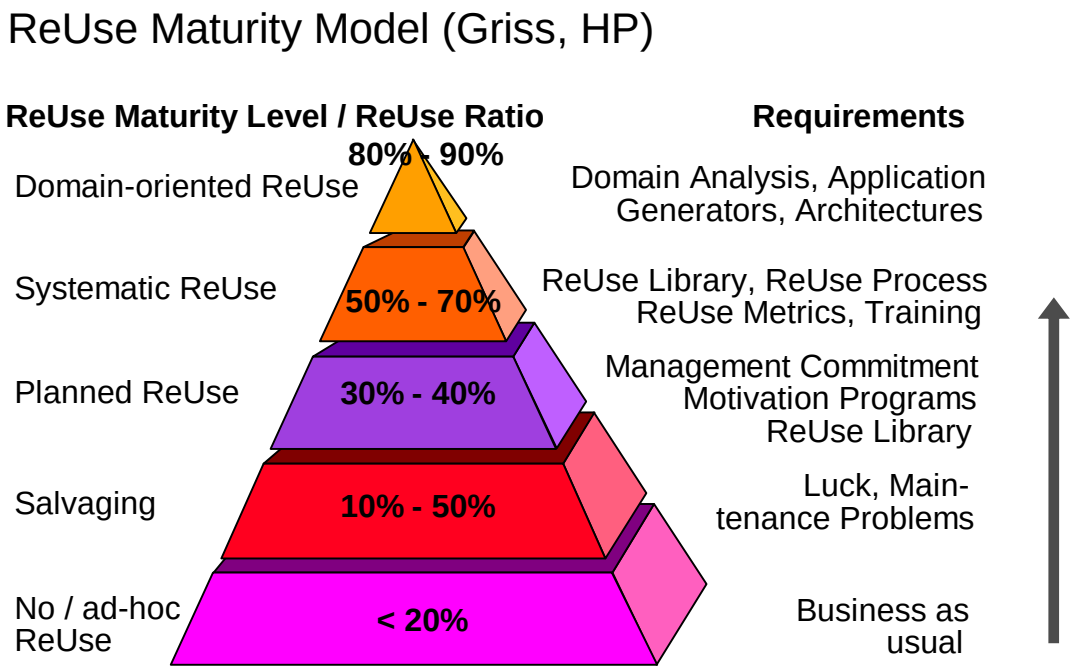
It was recognised that the full potential of productivity and quality improvement can not be achieved by ad-hoc reuse, but only through the development of a reuse culture emcompassing all hierarchical levels of the organisation and active daily practice.

The Impact and the Experience Gained

Technical Impact

Improvement of

- the software engineering process model
- the reuse organisation and infrastructure
- staff skills and motivation, management commitment
- application of standards
- SPC statistical process control / metrics
- reuse maturity (from ad-hoc to systematic reuse)
  - increased reuse ratio



  - improved reuse cost/benefit ratio
    - reduced costs through tool support for reuse activities (search, evaluation, classification)
    - increased benefit through better quality and reusability of the produced assets due to the application of standards, enabling technologies (methods and tools) and the methodical approach

Business Impact

Improved competitiveness through

- shorter project lead times, more precise adherence to time schedules
- improved quality (defect rate)
- improved productivity (function points / staff month)

all this leading to increased customer satisfaction and higher profitability

Key Lessons Learned

From the Technical Point of View

- Reuse need not be restricted to OO technologies. OO is neither necessary for, nor automatically assuring reuse and reusability.
- Reuse should not be restricted to code reuse. Reuse of other, more abstract workproducts of the life-cycle, such as design patterns or requirements, enable implementation-independent reuse on the one hand, and higher benefits when reuse can be started in early phases. Special interest should be given to reuse of experiences and know how. In that case, communication is the most important issue.
- It was recognised that not all of the issues involved, could be met by the adoption of state-of-the-art technology, standards and commercially available methods and tools. For reuse-related activities, such as reusability assessment, reverse engineering, re-documentation, domain analysis, etc. comprehensive offers can be found in the market, not so for direct reuse activities in connection with the establishment of a reuse library.
- Different technical reuse approaches (horizontal ⇔ vertical reuse, fine-grain ⇔ coarse-grain reuse, white-box ⇔ grey-box ⇔ black-box reuse,...) are possible. The right choice (or combination) is not only a technical question, but strongly influenced by the past and future business.
- Reuse is definitely contributing to software quality, especially to reliability, maintainability and portability. On the other hand, all of the quality factors, including functionality, usability and efficiency, influence the reusability of existing assets.

From the Business Point of View

- The full potential benefits of reuse in terms of productivity and quality improvement will only be earned, when applying a holistic approach, dealing with both, the technical and the commercial side of the medal, considering management, business strategy, corporate culture, organisation, training and motivation, capability maturity, key process areas, life-cycle, procedures, enabling technologies, methods, tools, metrics, risk and investments.
- The introduction of reuse is a non-conventional, but efficient maturity assessment method: hidden weaknesses become quickly apparent, especially in some closely related key process areas.
- Since the introduction of systematic reuse requires considerable up-front investments accompanied by several risk factors, an organisation should start a reuse program only

at a maturity level assuring that, relying on a sound product and process quality, produced assets are really worth reusing.

- The introduction of reuse requires management of change. Involved people, on all hierarchical levels, have to modify their working approaches, attitudes and general behavior. All requests to change meet doubts, reluctance or even resistance, especially when personel risks are involved („reuse is a job killer“). Therefore, high psychological and managerial skills are required.
- The possible benefits of development with reuse are correlated to the capability maturity of the organisation. Only when the organisation is able to produce assets, that are worth reusing, (or has already been in the past), benefits in terms of improved quality, shortened development times, reduced costs and risks, will be earned. Otherwise, the re-engineering effort will reduce the possible gains depending on the reusability of the existing reusable assets.
- The ability to yield the benefits of development for reuse is strongly related to business strategy. Only a clear vision about the characteristics of future development projects, make it possible for an organisation to make a sound economic decision about whether or not to invest in enhanced reusability of individual components or entire applications (development of application frameworks).

Consequential Organisational and Process Changes

The experiment shows, that productivity and quality improvement can be achieved through systematic reuse, provided, that all necessary preconditions are given.

The final results of the experiment are now being used to establish an outline plan for future improvements of the reuse process, covering e.g. iterative life-cycle models or the establishment of a permanent reuse organisation, and to carry out the reuse process improvement plan in the framework of the general process improvement plan.

Inter-departmental exchange of reusable assets, representing a hugh potential especially for large software producing organisations, requires organisational measures and very good software engineering practices from both, the producing and the reusing department.

Thus, the suggestion for further improvement in the outline plan for future actions will be the establishment of a central reuse manager and a reuse manager for every business area/field and project. The RAB (Reusable Asset Broker) function should be institutionalised, and complemented by a "ReUse Support Centre", covering the growing training and support needs, when more projects respectively business areas will adopt systematic reuse. The responsibility for the introduction of reuse in the different business areas should not be delegated to technical staff, but be directly taken by the responsible business area managers.

Whereas the market provides a wide range of code analysis/reverse engineering tools together with the respective support, training and consulting by the tool provider, the availability of tools supporting the management of reuse libraries and respective services is insufficient. Consequentially, the first working prototype of the reusable asset management system will be further developed.

Additionally, development with reuse needs a higher population of the reuse library, than the initial stock of reusable assets, that could be built during the experiment. Assets, especially high-quality assets, will be produced by either development for reuse or reengineering for reuse, including e.g. design patterns, as well.

References

[BOEHM88] B. W. Boehm, *A Spiral Model of Software Development and Enhancement*, IEEE, pp. 61-72, May 1988

[DIAZ85] Ruben Prieto-Diaz, *A Software Classification Scheme*, PhD thesis, University of California, Irvine, 1985

[DIAZ87] Ruben Prieto-Diaz and Peter Freeman, *Classifying software for reusability*, IEEE Software, pp. 6-16, January 1987

[JONES94] Capers Jones, *Becoming „Best in Class": the Path to Software Excellence*, in the Knowledge Base on the Software Productivity Research Inc. WWW Server, http://www.spr.com, Volume 3 Issue 1, March 1994

[MILLS87] Harlan D. Mills, Michael Dyer, Richard C. Linger, *Cleanroom Software Engineering*, IEEE Software, pp. 465-484, September 1987

[RANGHA57] S. R. Ranghanathan, *Prolegomena to Library Classification*, The Garden City Press Ltd., 1957

[SINDRE95] G. Sindre, R. Conradi, E.-A. Karlsson, *The REBOOT approach to software reuse*, in Journal of Systems and Software vol.30, no.3, pp. 201-212, 1995

Improving Software Development the Object-Oriented Way

Dr. Roman Cunis
MAZ Hamburg GmbH
Harburger Schlossstrasse 6–12
D-21079 Hamburg
cunis@maz-hh.de

Alexander Josub
DTK
Palmaille 82
D-22767 Hamburg
josub@dtkhh.de

Abstract

This paper focuses on the impact of object-oriented methods on software process improvement. It is shown that object-orientedness in itself is a paradigm that supports improvement to a degree, that other programming paradigms can't: "Objects" extend the notion of well-structuredness, replicability and reusability of results to software products themselves.

These insights are gained from a *Process Improvement Experiment* in the context of the European Systems & Software Initiative's (ESSI) "Software Best Practice" program. This paper is containing first intermediate results of the ESSI Project Nº21411, MAZ-PIE. The experiment is concerned with improving the software development process by using computer-based methodologies and tools for object-oriented analysis and design, reuse management, and configuration management. It is performed in the framework of the introduction of a company-wide quality system—based on ISO 9001 and oriented on the software engineering submodel of the V-Model.

This paper discusses the benefits of "objects" for software process improvement, gives a short overview about the underlying experiment, and discusses procedures to measure the results of the experiment.

Introduction

Process improvement—in the realm of software as well as anywhere—is about transforming some process that was always "somehow" working into a well-defined, well-structured, and well-documented process. This makes the underlying know-how transparent and thus accessible to everyone involved: the process finally will become replicable and incrementally improvable in itself.

With software development, the process under consideration has a feature that is not present in any hardware production process: The entire process is performed and all its products are created, manipulated, managed, and finally produced and packaged using the powerful capabilities of a computer. Thus it is intuitively appealing to manage and produce the by-products of a well-defined software development process—all those documents from requirements to integration test protocols required by software improvement schemes—in electronic form.

To do it manually—even using a computer to create the documents but decoupled from any development environment—is not only tedious but highly error-prone. Inconsistencies might evolve between requirement and specification documents and the software created thereafter, once software developers try to get the software actually running—and inconsistencies *will* creep in. Design reconsiderations during coding time are typically reintroduced into accompanying documents neither immediately (due to software production stress) nor afterwards (due to software maintenance stress). This will happen independently of the detail and accuracy of the definition of the process.

In the framework of the introduction of a company-wide quality system—based on ISO 9001 [[4]] and oriented on the software engineering submodel of the V-Model [[3]]—MAZ Hamburg GmbH decided to address these concerns by using computer-based methodologies and tools for object-oriented analysis and design, reuse management, and configuration management.

Object-oriented analysis and design has been selected not only because C++ is the premier language of development in the department that is performing the experiment. Object-orientedness in itself is a paradigm that supports software process improvement to a degree, that other programming paradigms can't: It extends the notion of well-structuredness, replicability and reusability of results to the software products itself. This aspect is the main focus of this article. It will be discussed extensively in Section 0.

Configuration management has been selected because CM is providing a set of features that contribute considerably to an improved process. It converts the heterogeneous world of electronic products in a software project (sources, data, executables, tools, documents, and so on) into homogeneously represented objects whose

interrelationships and interdependencies are well-documented. Process steps from requirement input to software changes to reintegration to reinstallation can be automated (to a certain extent). They become thus well-defined and replicable.
In order to evaluate the potential success of this approach to process improvement with respect to such features as

- applicability to the environment,
- acceptability of these tools with developers,
- actual support for the quality system,
- actual gain in efficiency and quality,
- measurable improvement of the software development process,

a *Process Improvement Experiment* has been defined in the context of the European Systems & Software Initiative's (ESSI) "Software Best Practice" program, and the proposal has been accepted. This paper is containing first intermediate results of the ESSI Project Nº21411, MAZ-PIE [[2]]. Section 0 of this paper will give a summary of the project. Section 0 discusses the approach on measuring success.

Objects for Software Process Improvement

Object-orientedness in itself is a paradigm that supports software process improvement to a degree that other programming paradigms can't: It extends the notion of well-structuredness, replicability and reusability of results to software products themselves.
This bold thesis is derived directly from properties that are typically attributed to object-oriented software development. This section is explaining the thesis theoretically from the "object" point-of-view. The process improvement experiment presented in Section 0 is performed to evaluate the thesis and its consequences practically.

Properties of Objects

First, those aspects of objects are briefly lined-out that characterise objects best and are most important to prove the thesis. Objects are typically described by the following properties:

- Coupling of function and data
- Encapsulation
- Classes
- Inheritance and specialisation
- Polymorphism and generalisation
- Interrelationships

Each of these properties is discussed in turn:

- In an object its data (local variables, attributes, properties, state variables) and the functions operating thereon (member functions, methods) are tightly *coupled*. Every member function of an object has immediate and exclusive access to the object's data. Even if several objects exist of the same type (class, see below) it is always guaranteed that the function code—though it is physically existing only once—will access the right data. In this respect object-oriented programming is combining the best of functional programming (i.e. locality of operation, no inadvertent side-effects on shared data) and procedural programming (i.e. ability to store state on persistent data areas).
- These properties are additionally supported by *encapsulation*: local data and auxiliary methods can be completely hidden from users of an object. These users have only access to an object's public methods—the rest is implementation detail. This supports a high decree of locality of modifications. Traditionally, all local data is private (hidden), and all methods of an object are public. Some OO languages allow more fine-grained encapsulation: i.e. specifying the degree of visibility for every attribute and method individually. The set of all publicly visible method is constituting an object's *interface.*
- *Classes* define the characteristics and the behaviour of an object, from a programming language point-of-view they define its type. They define which local data and which methods are present as well as their encapsulation attributes. The implementation of a method is also associated with an object's class, thereby guaranteeing that an object's method code is properly shared between all objects of a class.

- *Inheritance* is one of the two most powerful properties of objects. Classes can be defined as specialisations of one or more other classes (base classes, super-classes), thereby inheriting all of the definitions already present, i.e. all local data and all methods. Moreover, the new subclass (derived class) may add its own definitions and extend or override inherited methods. In a family of similar objects shared properties can thus easily be identified and effectively localised, while individual extensions can equally well be added where appropriate. Often the similarity does only comprise identical interface methods without any internal detail: such base classes are typically called "abstract". Subclasses provide implementations in different variants. Even more important: specialisations can be added at any time without changing the code of base classes.
- The twin brother of inheritance is *polymorphism*. In every context where a certain base class is used through its public interface, any concrete object of any subclass can actually be substituted. The general calling context will automatically use the most specific method definitions. This makes the type of extensible frame-works possible that have become a hall-mark of object-oriented systems. A general environment for specific problem types can thus be provided for different applications that can easily augment it with their specific objects.
- Objects are individuals. They are passed around not by copying their data but by using unique references (a name, a pointer) to an object. It is thus very easy to express complex *relationships* between objects of different classes. These relations provide a powerful way to mirror complex interdependencies, like usage, containment, partitioning, association—to name only a few of the more general. Every application can easily add its own relations.

A good general overview on object technology can be found in [[1]].

Ubiquitous Objects

Objects are everywhere. Not only given a "horizontal" view, that is that nearly everything in an application can be viewed as an object. Also in a "vertical" view on the different levels of software design objects can be found on every level.

- On the lowest implementation level, objects represent *abstract data types* by virtue of their encapsulation and interface capabilities.
- In an object design, every program entity can and should be viewed as an object that is a bit more complex than a basic data-type (e.g. integers[28]). The properties described above are thereby added to everything in a design, and will thus greatly extend its extensibility, modifiability, and reusability.
- Objects constitute *modules* by means of their encapsulation and data-and-function coupling properties. Objects may contain objects, and the "outermost" object may well represent an entire block of functionality. Having objects as modules allows for easy replacement and variable specialisation of modules thereby exploiting the object's inheritance property. Moreover, such modules can effortlessly be duplicated in an application without the danger of encountering clashes in global namespace.
- Objects can be *processes*. In a multi-process environment every process is represented by an object and is thus automatically encapsulated—and again extensible and easily exchangeable.
- Objects can represent *real-world things*. Requirement analysis for an application will identify the things that are to be considered in the system, things that act upon other things, things that interact with other things. Each of these things can be captured as a class with attributes and behaviour, thus bringing the benefits of OO to the requirements level.

Object-oriented Analysis, Design, and Reverse Engineering

These features are now reviewed from the perspective of the software development process. A well-defined process starts with requirement analysis, proceeds through several levels of increasingly finer design until modules can be implemented. The results have to be integrated again over several levels of decomposition until the overall system finally comes into being. On every level of integration one might run into difficulties and reiterate back to the appropriate level of design decomposition.

If objects are used, requirements analysis is already providing a bunch of objects of the application's real-world. There are different strategies how to identify these objects. Several authors propose to simply take the requirements' textual representation and turn every noun into a class and every verb into a method of some appropriate class. Budde et.al. [[8]] suggest to identify the tools (things that act on other things) and materials

[28] Some OO supporters do even count these as objects—though I personally doubt the individuality of—say—the number "7".

(things, that only are acted upon) in an application and turn these into two categories of objects, where tools typically have behaviour and materials have state.
Anyhow, for each of the objects and classes identified, their possible states have to be specified and the conditions and means under which state changes may happen. Responsibilities of every object are to be determined, which tasks it may perform, which services it might provide, and which services it might need from other objects.
Decomposing a complex system in this way greatly aides the analysis process: thinking about interaction between objects is thus separated from thinking about how objects should achieve their tasks.
The result from this process will be a static object model—comprising class inheritance hierarchies— and a dynamic model—defining the ways of interaction between objects of different classes using state diagrams for the inner dynamics of an object and interaction diagrams (e.g. use cases) for the collaboration and co-operation of objects with other objects.
The great thing about using objects for analysis is that design does not need to transform what is already acquired into procedures and data. Instead, design takes the model from analysis and incrementally puts more and more detail into it: local attributes, auxiliary methods, and other design-level objects representing modules, processes, various entities and data types. There is a smooth transition from analysis to design that can't be found elsewhere. An additional benefit is that design decisions can easily be communicated back to customers, because they are expressed in the same model that resulted from analysis and had been discussed with them in the first place.
This smooth transition continues all the way to implementation, because an OO design can isomorphically be mapped to most OO programming languages chosen.
Using an OOA/OOD tool one gets a single repository stuffed with all the information about all the objects. Analysis, design, and even coding can be performed in one electronic model of the application. All necessary documents with every desired level of detail can be generated from the model as well. (Cf. Figure 1.) If there are ever changes to the model on a lower level that affect a higher level feature, i.e. changing specification results during design, there is no need to re-transform these changes into a higher level representation: the change would be immediately visible.
This is also true—at least to some extend—for changes to the design that are brought upon through changes in the code. A process that is called "reverse engineering" reinterprets the code and reconstructs the object model from the code. The results are then merged with the existing model and differences can easily be found or directly be updated.
Reverse engineering also provides the means for entering into the world of OOA/OOD tools even if object-orientedly coded software is already existing. It will extract as much information from the code into the model, thus leaving the software engineer with a raw model that can easily be edited to reflect the underlying analysis and design.

Objects Improve the Process

Using an OOA/OOD tool as described above supports a well-defined and replicable software development process. Simply by using a tool that enforces a certain level of detail, a certain level of conformity, and a high level of consistency between submodels, well-structured analysis and design results and documents can be produced.
This can of course be achieved with any conventional analysis and design tool, too. The difference lies in the absence of transformation rules between analysis, design, and coding. Having changes in the lower levels be immediately reflected in the higher levels greatly improves the consistency between documents and models, and thereby the quality of the overall documentation.

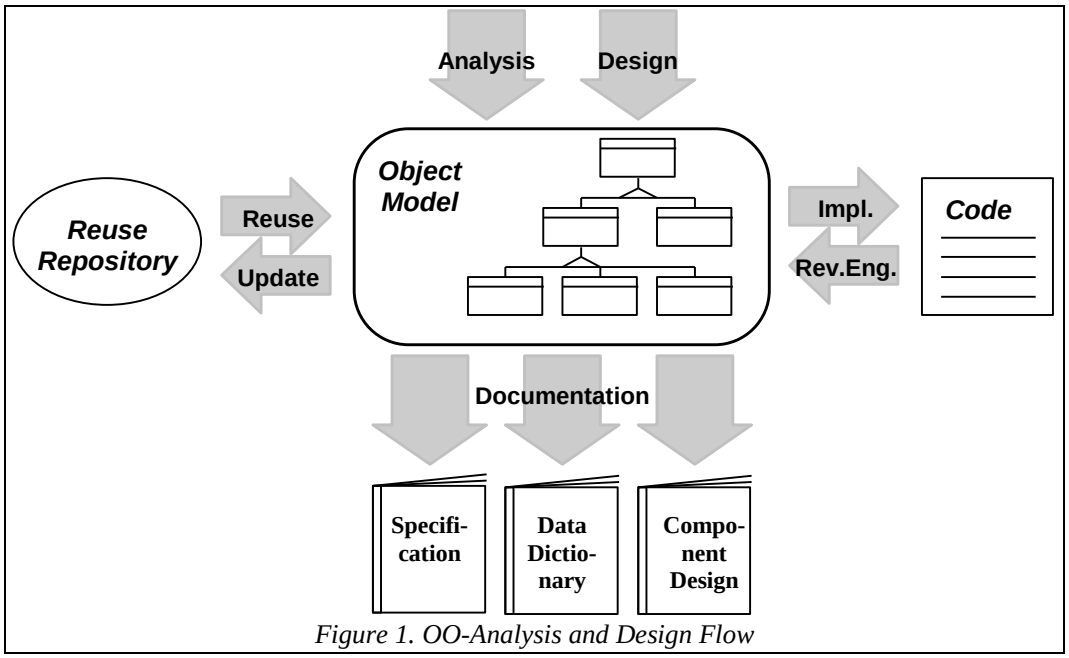


Figure 1. OO-Analysis and Design Flow

This leaves us with the thesis, that with objects the notion of well-structuredness, replicability and reusability of results can be extended to software products themselves. The magic answer to this is *Reuse*. Object-oriented designs provide a level of reusability that can not be achieved by conventional libraries of functions.
Reusing classes does not only provide functionality, but predesigned data-structures and interfaces as well. Best example for these is the ever-growing family of container classes that provide a wide range of well-designed and often-used data structures together with efficient algorithms to use with them. And these classes can not only be used as they are—due to inheritance they can be specialised and tailored to the needs of an application.
Reusing a class *framework* does not only mean to call into it, but to integrate with it. Using the means of specialisation in the application and of polymorphy inside the framework allows to link with the framework in such a way, that complete problem-specific kernels can be executed within: The application provides implementations tailored to its needs by inheriting from predefined classes. A good example for these might be a simulation framework: an application provides the definitions for its real-world objects by specialising appropriate abstract classes inside the framework. The complete simulation engine including control, statistics and visualisation will automatically run with the applications' objects.
Reusing existing class libraries and frameworks makes well-defined software available that has a replicable structure, as the same building blocks inherited from the libraries are used over and over again. Reuse is not only a matter of implementation but already a matter of analysis and design. If the classes and frameworks to be reused are modelled completely inside the OOA/OOD tool and its repository, they can easily be integrated (static and dynamic model and all) into a new model.

Reuse Management

Finally, to have software results reusable and thus replicable, the application-specific parts must be designed in a reusable way, too. This is the hardest part. To identify a subcomponent of a system as candidate for reuse and to design it accordingly needs a certain amount of experience with object-oriented concepts. Though often promised, just by designing a subcomponent with classes and objects doesn't achieve reusability automatically. Care must be taken that as few application-dependencies as possible appear in (generally abstract) base classes, and that their interfaces are designed with enough foresight to take potential future requirements and extensions into account.
Reuse must be managed, and reuse management should be built in into the software development process. Unless all developers are OO specialists, a *reuse manager* will be needed. The reuse manager should take care that available class libraries and frameworks are actually reused, and that new designs are designed for reuse wherever possible. Two new steps have to be introduced into the development process: *reuse planning* and *testing for reusability*:

- *Reuse planning* should take place immediately after the first design step. The design should now contain all important classes and their interrelationships, and subcomponents have been identified. In this situation the reuse manager should

  - check the design whether components available for reuse have actually been used wherever appropriate: if not he should propose design modifications so that reuse is increased,
  - inspect the design for candidates for future reuse: if suitable components can be identified he has to make sure that the design is modified so that reuse will actually become possible.
- *Test for reusability* should take place during integration, when reuse candidates are completely and successfully integrated. The reuse manager has to take a candidate component, test it for reusability, make sure that separate documentation sufficient for reuse has been prepared for the component, and finally add the component to a reuse library.

Note that this will actually increase the efforts for the current project. The team has to be truly committed to reuse to accept the additional burden. Investment into reuse is an investment into future projects. Consequently the effort is only appropriate if future projects are planned in which the components will actually be reused. Even with good foresight, one should expect that a component designed for reuse will still have to be redesigned the next two or three times it is going to be used until it is a truly complete library or framework—especially if the functionality therein is a bit more complex than a container library.

OOA/OOD Methodologies

For a couple of years now, OOA/OOD methodologies have sprung up in a rate greater than 1 per year. What has come to be known as "the method wars" raged on. To name only a few of the more prominent:

- Booch
- Rumbaugh's Object Modelling Technique (OMT)
- Jacobsson's Use-Cases
- Fusion
- Coad/Yourdon
- Shlaer/Mellor

About two years ago, Grady Booch started what he called "unilateral unification": he incorporated what was better in other approaches into his own method [[6]]. In the consequence, James Rumbaugh agreed to co-operate with him on the development of a unified methodology. In 1995, Ivar Jacobsson joined the two. The method currently known as the "Unified Modelling Language" (or UML for short) has been pre-released in version 0.9 in June 1996, the first release Version 1.0 is announced for the beginning of 1997 [[7]].

With the advent of UML supported by the three most prominent men in the field, the method wars are considered to be over. I expect UML to evolve into a widely accepted standard in the next two years.

Most tools that up to now supported either Booch or OMT do meanwhile support what is already known of UML—or have announced to support full UML shortly after its complete release.

For the Process Improvement Experiment reported in the next section, it has been decided to use Booch as available and switch to UML as soon as possible. The reason for this is that—in the opinion of the authors—Booch is closer to C++ design than any other method.

The Process Improvement Experiment

Project Overview

MAZ Mikroelektronik Anwendungszentrum Hamburg GmbH is a development and system house founded in 1990 by the Free and Hanseatic City of Hamburg. The main objective of MAZ is product-oriented technology transfer in the area of microelectronics applications. Software development plays a significant role to reach this goal.

MAZ has undergone a BOOTSTRAP assessment in December 1994. As a result of this a quality system based on ISO 9001 is currently being installed. Its software development guidelines are strongly based on the software engineering submodel of the V-Model, which will improve the software development process significantly.

During the Process Improvement Experiment methods and procedures for object-oriented analysis/design (OOA/OOD) and configuration management (CM) as well as appropriate tools will be introduced into the software development process. The overall goal of the experiment is to increase the maturity level of the software development process with respect to requirement analysis and definition, architectural design, as well as configuration management and change to 2.5 up to 3 according to the BOOTSTRAP [[5]] ranking. If the PIE is successful these techniques will be trained and used throughout the department.

The experiment is carried out in five phases. During Phase I organisational issues of the project have been settled. Phase II dealt with the selection of appropriate methods and tools. A first introduction of the selected methods and tools in context of a strategic baseline project is currently executed in Phase III and evaluated. Phase IV will be concerned with a broader introduction. The latter two phases comprise extensive training for

the staff members and augmentation of the software production guidelines and the quality manual of the company. During Phase V a BOOTSTRAP assessment will be carried out in order to evaluate the results of the experiment.
DTK GESELLSCHAFT FÜR TECHNISCHE KOMMUNIKATION MBH is a software consultant and accredited BOOTSTRAP assessor. DTK will assist in the definition of suitable metrics, continuously monitor the process improvement with respect to its maturity level, and aide in the development and execution of training procedures regarding these metrics (see Section 0).

The Baseline Project

The baseline project chosen is concerned with the development of one of the strategic products of MAZ. The system is a knowledge-based machine fault diagnosis and maintenance support system, known as DiaMon (*Dia*gnosis and *Mon*itoring).
The system consists of

- a specially developed digital signal processing board for sensor-data pre-processing,
- a real-time monitoring package for evaluation and long-time storage of sensor-data, including fault detection mechanisms based on a fault-tree model, and fault signalling mechanisms to whatever control-unit is around,
- an interactive, windows-based front-end for visualisation of sensor-data and interactive guidance through the fault-detection process if user-interaction is required (e.g. manual probes),
- a powerful editor for generating fault-tree and machine models which are stored in a database.

The software for this system is developed in C++ using the object-oriented paradigm. The system is not yet completed due to constantly changing requirements, a significant part of development has been performed in the manner of rapid-prototyping. After a 12 man-year effort (with at least 9 more man-years to go) the software consists of roughly 120 modules, 20 percent of which are multiply reused in different software configurations. Early, yet incomplete versions have already been delivered to customers.
The process improvement experiment is focusing on that part of the software that is already multiply reused under different configurations. This part would benefit most and has the most significance for the successful evaluation of techniques and tools.

Adapting the V-Model

As stated above (cf. 0), it is planned to incorporate the experiences gained from the project into the MAZ's software guidelines during the course of Phase IV. As these guidelines are based on the software engineering submodel of the V-Model this will constitute an adaptation and augmentation of that part of the V-Model to the needs of object-oriented software development (or at least the MAZ's brand thereof).
The V-Model is effectively prescribing an incremental, top-down software development process with provisions for cyclic refinement. Thus, a development process centred around an object repository is fitting well into that framework.

- During *Requirements analysis*, requirements from an application's point-of-view. Requirements will be stated in the terms of use-cases and entered into the object repository. Requirements will typically not define any objects or classes, as these do already incorporate design decisions.
- During *Specification*, top-level class diagrams and the responsibilities, interfaces, and dynamics between top-level classes are defined. The assignment of classes to modules is corresponding to the definition of so-called software configuration units (SWCU), the V-Model's term for the main building blocks of a software project.
- A new role is introduced: the *Reuse Manager* (see 0) has to perform the new activity "reuse evaluation". SWCUs are analysed and evaluated with respect to the reuse repository in order to answer the following questions:
    - Can the SWCU be replaced by or at least incorporate a reusable component from the repository? Can the specification of the SWCU be changed so that reuse becomes possible?
    - Is a SWCU a promising candidate for later reuse in other projects? Is it designed for extensibility and reusability? How can its specification be changed, so that the class hierarchy does properly reflect application independent aspects at its base, and application dependencies in its derived classes?
- During *Coarse Design*, the static and dynamic models of all modules and SWCUs are completed.
- During *Fine Design,* module-internal auxiliary classes and objects are designed and incorporated where necessary. The exact interplay of all objects is determined.

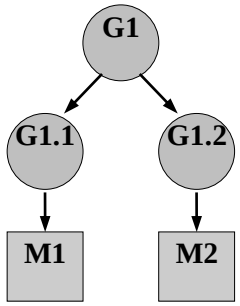
- In *Implementation*, the overall code structure of classes and their methods will be generated automatically from the object repository. The internal workings of algorithms and methods have then to be filled in.
- After *Integration* of SWCUs and components, another new activity is introduced for the reuse manager: "Updating the Repository". In this activity SWCUs designed for future reuse will be reviewed under this point-of-view and—if found satisfactory—added to the reuse repository.

The V-Model amply provides pattern documents for all relevant documents to be generated during the process. These patterns will be adapted to reflect the object-oriented structure of the software, i.e. wherever a structuring is given along the line of functions and procedures, this will be replaced by classes. As classes combine data and interface, the documents named data dictionary and interface catalogue, respectively, will be combined to one object dictionary document.
The OOA/OOD tool selected (and several other of the major tools, too) provides the possibility to actually augment an electronic pattern document in such a way, that all relevant object data can automatically be filled in from the repository. The appropriate definitions and filters will consequently be defined in all pattern documents to extract the proper information wherever possible.
One final remark: A revised version of the V-Model itself is currently under preparation by its authors and will be released in the first half of 1997. This will incorporate—among other things—provisions for the support of object-oriented software modelling. If possible and necessary, the MAZ's software guidelines and pattern documents will be kept in sync with this development.

Measuring Success

The results of such an experiment are inherently difficult to measure. Most of the benefits will typically not turn out immediately after introduction of tools to one project. To the contrary: the project used for the experiment might actually fare worse because of the additional burden in getting the tools to interact smoothly with what is already present. Most benefits will be felt in subsequent projects when results might actually be reused or when software changes can be performed more efficiently.
To obtain an overall quality measurement, the MAZ department involved in the experiment is assessed twice using the BOOTSTRAP evaluation technique [[5]]. One assessment took already place in December 1994 and finally prompted the efforts to perform the Process Improvement Experiment. The result of the assessment has been that the current status of maturity level is 1.25. This result is in line with the average maturity of the European industry, and reflects the difficulties that small companies face in establishing a comprehensive suitable quality system. At the end of the experiment, the department will be subjected to another BOOTSTRAP assessment. The overall aim of the experiment is to raise the maturity level to 2.5 to 3.
To acquire suitable base data to perform this evaluation and to measure improvement results concurrently during the experiment, a goal-driven approach based on the Goal-Question-Metric (GQM) method and the results of the ESPRIT project Application of Metrics in Industry will be applied during the project. This method has been selected because of its flexibility and adaptability to any type of organisation and metrication objective, and because it is based upon good sense and logic.
According to the GQM-method a starting assessment and an analysis of the process are done first. The following step leads to the specification of the metrics and to the definition of a data acquisition and measurement plan. Primitive measurement data are periodically collected and analysed, according to this plan. During the final step the measurement data are reviewed and distributed. Then the metrics are validated and related to the initial goals. Corrective actions are subsequently implemented and new goals are defined.
The method starts with the assessment of the project environment and the definition of the primary goal(s). This step has already been taken with the formal BOOTSTRAP assessment prior to this project, as mentioned above. Consequently, the overall management goal is also defined in terms of a BOOTSTRAP ranking.
Subsequently, primary goals are analysed and broken down into sub-goals, thus building a hierarchy of goals. A small work group will analyse the primary goals. The group consists of a metrics promoter and the managers and engineers who are affected by the goals. Each primary goal is analysed and broken down into sub-goals to create a goals tree. A table of questions is created, that documents the thinking process of how each branch was derived. Final questions at the end of each branch are a specification for the metrics.

One of the main objectives of the analysis is to set goals that correspond to domains of responsibility and to the decision making activity of teams or individuals. The measurement data that is collected will then support decision making and goal achievement. The goals tree is used later when the measurement data is exploited.
A second objective is to achieve greater precision in the primary goals. A goals tree can be used to trace through from customer quality requirements to processes that affect that quality aspect. It can be used to trace complex entities back to constituent entities and attributes.
Once created, the goals tree is checked for consistency of sub-goals with primary goals and of goals with decision making. Figure 1 shows an example for a goals tree, as currently developed in the project for the area of configuration management.



G1 Improve the software development process w.r.t. to CM to a minimum of 2.5 according to the BOOTSTRAP ranking
G1.1 Increase the number of elements under CM control
G1.2 Redurce the effort for CM

Figure 2. Example of a goals tree.

After validation of the goals tree a measurement plan will be written, describing mechanisms for data extraction, and naming responsible persons and the time frame of the action. The plan is written by the metrics promoter and validated by those involved in the definition of goals. From the metrics specification a complete definition of the metrics and the analysis and collection procedures are produced. A metrics template will be used to support this process. Collection forms and simple tools will be used to aide in the process. Once the plan has been accepted, primitive data will be collected and subsequently verified for completeness and accuracy.
Finally, primitive data will be merged into measurement data to be evaluated. At this stage an effective presentation of the measurement data will be given and shared with those involved in the relevant goals. The first aim in goal-oriented measurement must be to report measurement data to those participants whose particular goals are affected by the data. A statistical technique or model is only to be used if appropriate and with care.
Measurement data will then be analysed. Analysis of data with goal-oriented measurement is self referencing and fairly simple. For example, estimates use past data, error rates should tend to zero, progress should tend to 100%, and so on. However the analysis must always be done with careful reference to the context.
As already mentioned, this measurement process will be finalised by a complete formal BOOTSTRAP assessment at the end of the project. It will give an overall view of the improvements hopefully achieved by even this rather short experiment. A plan of future actions and process improvement activities will be defined beyond this project, considering the results of the measurement activities and on the basis of the final assessment.

Conclusion

This paper described early results from ESSI Project 21411, MAZ-PIE, in which OOA/OOD as well as configuration management methodologies and tools are evaluated with respect to their impact on process improvement in the course of software development. It has been shown that object-oriented methods are particularly well-suited for this task, because they constitute a paradigm that supports software process improvement to a degree, that other programming paradigms can't. It has been argued, that in order to reap the benefits of object-oriented reuse the software development process—as e.g. described by the software engineering submodel of the V-Model—has to be extended by the role of a reuse manager and his activities. Aspects of the V-Model that need to be adapted to object-oriented development have been discussed.
Finally, it has been shown, how improvement can actually be measured in an area where the results of such an experiment are inherently difficult to measure because most of the benefits will typically not turn out immediately after introduction of tools to a project.
This paper reflects the results of the PIE project at the time of preparation of the paper. This has taken place considerably earlier than its actual publication and presentation. As the project did only commence in April this year, detailed results of the project could not yet be reported. The authors will be able to present more detail on the conference itself and in later publications in the course of the project.

Once MAZ-PIE will have been concluded successfully, MAZ Hamburg GmbH plans to gradually introduce the methodologies and tools evaluated into all departments involved with software development. To this end, the results of MAZ-PIE will be disseminated on in-house workshops. Use of the tools will be trained, and special consultancy sessions will be provided to introduce the tools into existing projects. Other aspects of object-oriented software development, like e.g. object-oriented real-time development or object-oriented testing, will be evaluated and considered for introduction in due time.

References

[1] David A. Taylor: *Object-Oriented Technology, A Manager's Guide.* Addison-Wesley, 1990.

[2] Roman Cunis, Heinz Marburger: *ESSI Project 21411, MAZ-PIE, Project Programme.* MAZ Hamburg GmbH, 1996.

[3] *Vorgehensmodell (V-Modell) zur Planung und Durchführung von IT-Vorhaben.* Bundesministerium des Innern, KBSt, Bonn, August 1992

[4] *ISO 9001 : 1994, Quality systems — Model for quality assurance in design, development, production, installation and servicing*

[5] P. Kuvaja et.al.: *Software Process Assessment & Improvement. The BOOTSTRAP Approach.* Blackwell Publishers, UK, 1994.

[6] Grady Booch: *Object-Oriented Analysis and Design, 2nd Edition.* Benjamin Cummings Publ., 1994.

[7] Grady Booch, James Rumbaugh et.al.: *The Unified Modeling Language, 0.9 version addendum.* Rational Corporation, 1996.

[8] R. Budde et.al.: *Tools and Materials: an Analysis and Design Metaphor.* In: Technology of Object-Oriented Languages and Systems TOOLS 7, Prentice Hall, 1992

Object Oriented Modeling of Work Processes

Kurt Walk
walk@via.at

Richard Messnarz
rmess@genrix.ie
http://www.iol.ie/~iscn/info

Abstract
Work processes are viewed as the behavior of systems of asynchronously cooperating objects interacting with people. A lean architecture of object systems is presented and is shown to be adequate for the modeling of software development processes. This includes the modeling of user roles, of work places and authorization, of information management, and of process control. Interleaved and cooperating processes, processes managing other processes, and the control of change can be included in this same basis in a consistent manner. Experience gained in a case study is described.

Introduction

The formal modeling of work processes is gaining increasing significance for the analysis of organizations with respect to their efficiency and the quality of the products delivered and services rendered. Besides being the basis for the comparison of organizations and for planning improvement, process models allow insights in the roles of people in an organization, and in the view participating people may have of their work environment. An adequate notion of 'process' should therefore be selected with care, in order to provide a common, firm basis for modern analysis of work environments.
The analysis of work processes usually goes together with the analysis of the information system services required to support the processes. Services may provide support at different levels. In the order of increasing central significance for the management of processes these are:

1. the support of production activities like: writing letters, debugging programs, managing business data (production support)
2. the support for planning and controlling work like: setting up production plans, comparing actual data with plan data, managing personnel data
3. the actual control of work by: distributing work requests, automatically providing level 1 and level 2 services where needed, monitoring completion, signaling out-of-line situations (process support).

The development of information system services is itself a rather demanding work process. It is, actually, more complex than many other types of processes like, for example, administrative processes which need not allow for the dynamic creation of new work, accommodating changing assumptions and partial rework, and unplanned intervention. Software development is the area chosen in the following for discussing the nature of work processes and for drawing examples from.

Many process modeling languages are proposed and in use, many of them also executable by process support environments ([Hump89], [CuKO92], [ArKe94], [FiKN94], [Warb94]

[ScPo95]). They are mostly variants of notational systems which express process models as relationships among: activities, work products (artifacts), roles of people, tools, and other auxiliary process elements.

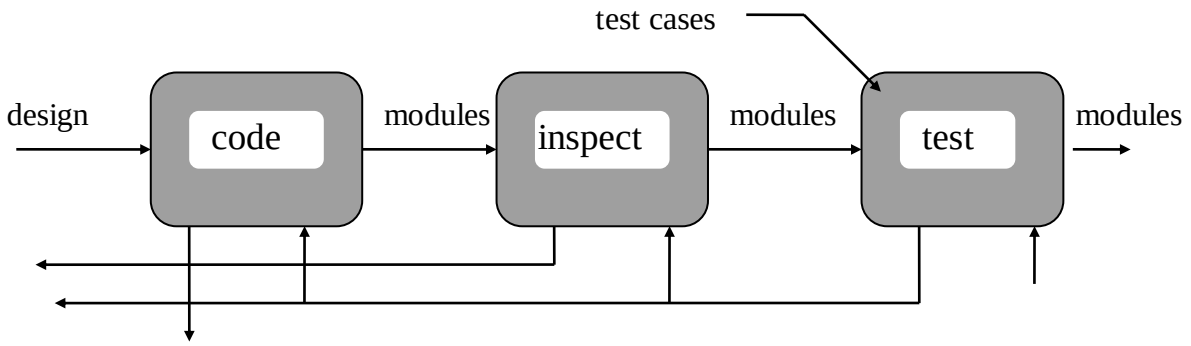
The approach described in this paper is intended to serve as a guidance for the analysis of work processes and specifically aims at a concept of models which structurally reflect the real world work environment. It is based on the view that processes are the behavior of systems interacting with people. Its focus is the system architecture. People are outside the system - there is no attempt to model people. People interact with the system through interface components called work places. The type of a work place characterizes the role its user can play with respect to the system. Work places contain the available tools, they provide access to common data, and communication links to other work places. They also are the repository for information specifically addressed to their users, like work requests, work objects, or mail.

An architecture of asynchronously cooperating objects is used. No features of a process modeling language are being proposed here, except that certain standard graphics is used in the paper for representing objects and their relations. The modeling process itself can be guided by published object-oriented methodology, in particular by the approach advocated by Jacobson [Jaco92].

It is mandatory for the success and a reasonable return on investment of a process modeling project that people actually participating in a work process also cooperate in establishing and reviewing the evolving models. To do this, they must find their work environments presented in a way they can identify with. This is particularly important for smaller and middle-sized organizations, which often start from a status of having no formally defined processes at all. Experience gained in a case study with a middle-sized Austrian company is reported.

Characteristics of Software Development Processes

We begin the discussion with a simple sub-process in the development of computer software (a simple variant of the example in [ISPW-6]), drawn as an activity network (see e.g. [Hump89], [Chro92]):



The boxes ‘code’, ‘inspect’ and ‘test’ represent the activities of, respectively, coding, inspecting, and testing software components, called modules. The activities take inputs (design, modules, test cases), and produce output (modules). These are the work objects of the process.

This activity based model conveys the following rules to a human reader: software components are coded based on design input, they are subsequently inspected and tested. Inspecting must not start before coding is complete, and testing must not start before inspecting was done and test cases are available. If problems are found in an activity, it may be necessary to recur to earlier activities, for example: to go back to coding if problems are discovered during inspection.

This example exhibits some general characteristics of most work processes:

- certain activities have to be performed in a certain sequence
- there are points where activities need to be synchronized
- production activities are often followed by check and repair activities, which causes loops in the process.

Further analysis, however, leads to many more questions and answers which one should be able to enter into a complete process description. It reveals more characteristics of work processes:

The role of people

Activities are performed by people in different roles. Roles differ in the type of activities, the services used, the required access to information, levels of authorization and, of course, they require different skills. The specification of the roles of people participating in a process must be a part of the process models.

People in general participate in several processes and they are related to one another in various ways. Relationships among people should also be seen from a process point of view: the essence of the manager-to-employee relationship is the existence of processes like work allocation, performance appraisal, salary determination, reporting, etc. The participation in a team is another type of relationship, which means participation in the same production process, common reporting and meeting processes, etc.

The total of established capabilities, responsibilities and relationships of a person we call a ‘work place’. An organization is characterized by the total of its work places. We need ways to represent work places in the process models.

Work assignment

Activities need to be assigned to people (or assumed by people) in conformance with the process rules. Per the above example, test activities should not be performed before inspection is complete. It turns out, however, that this description lacks precision. We can interpret it in at least two ways:

1. all inspection activities must be complete before any testing can start. In this case we talk about a ‘checkpoint’ in the process
2. any module may immediately be tested after it has been inspected. This is a rule established for the work object ‘module’.

Rules established for work objects are often founded in the technical properties of the work objects. It is, for example, a technical property of a piece of source code that it must first be compiled before it can be executed. We can describe work objects in ways which do enforce

these technical rules.

Change

Development processes are accompanied continually by change. There are backward change requests, caused by the discovery of problems in checking activities. These change requests may ask for a change of work objects, like software components, which already exist, or may require the creation of new work objects. They may also cause a change in the work objects created in earlier phases, like in design objects if design errors are detected later in coding and testing activities. In any case, development is not characterized by a fixed number of pre-planned activities, but work is created dynamically during process execution. In many cases, work necessary to be done is presentable by the work objects themselves: a software component which is yet untested gets the status 'untested' and thus presents a piece of work to be done, namely testing the component in order to bring it to the status 'tested' and thereby releasing it for execution.

There are also forward change requests, caused by new requirements or new insights, which do have an effect on work already done. A changed design may invalidate pieces of code already produced. A change process must determine work objects still valid, new work to be done, and work to be repeated. If work to be done is represented by the work objects themselves, this is a manageable task. The status of some of the objects may need to be reset, and new objects representing new work may have to be introduced through a work place with the appropriate capabilities and authorization.

The status of an object is the condensed history of what has been done with it. For continuing a process it is (within limits) not relevant whether a work object has been set to 'invalid' just once, or several times. Relevant fact is: it is invalid and needs to be redone or removed. We can speak about 'history hiding' by analogy with 'information hiding', in that irrelevant parts of history are hidden in the same sense as irrelevant properties of (the representation of) information are hidden [HuKe89].

Interacting processes

A person in general participates in more than one process at a time. Besides doing testing work, a person may participate in planning activities, status reporting, education, correspondence, or in some other project. Similarly, work objects may take part in more than one process. For example, a piece of design while being used in development may participate in a library back-up or in a measurement process.

Our goal is to develop a consistent framework for representing all the processes, instead of describing just isolated dimensions, like the organizational dimension, the data dimension, or the tools dimension. In particular, change processes are of the same nature as other work processes, though they may involve people in roles different from roles in a production process. The same holds for system management processes, responsible for the setting up and the tuning and change of system components.

Controlling processes

It is part of our objectives to envision level 3 process support, i.e. the computer aided control of work processes. If not for actual realization, this is a test of the precision and lack of ambiguity of a process description.

The design of process control should take the views of participating people into account. People are not just processors put under the control of an automated agent, but they play an active and creative role. It is a property of the work places of people whether they allow for adequate freedom for their users. Object-oriented work places can be designed to be populated with work requests or work objects representing work to be done but leaving room for personal choice and planning of work.

The above characteristics suggest process models based on active, asynchronously communicating objects. Objects can represent the work objects of concern, the work places, and controlling agents. They play an active role and thus assume ‘responsibilities’ in a system. Asynchronous communication in any case is mandatory for the modeling of processes involving (asynchronously) acting people.

An Object System Architecture

A process, in the more formal sense, is the behavior of a system. A particular notion of ‘process’ is based on the architecture of the underlying system. We are using an object system architecture as the base.

Properties of a process should be modeled consciously. They should not be implied, or even obscured, by properties of the underlying system. Therefore, the architecture of the object system, and the notational means, are intentionally kept lean at this point. More properties and notation can be added as demanded by practical use, but extensions should be defined in terms of the basics.

An object system is characterized by: a set of object classes, a set of (actual) objects, each belonging to one of the object classes, and an asynchronous communication system.

Objects

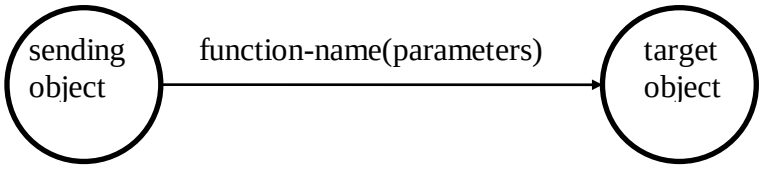
An object has a unique name, actual data, and is the member of a class. Objects communicate by sending each other stimuli via the communication system.

Object classes

An object class has a unique name and is characterized by a set of (partial) functions. The functions transform object data and parameters (the domain of the functions) into object data and the specification of a set of stimuli (the range of the function).

Stimuli

A stimulus is specified by a function name, the unique name of the target object, and parameters:



Create and destroy

There are two special stimuli:

create(class-name, parameters, name-of-sending-object)

and

destroy(name-of-object).

'create' creates a new object of the specified class and returns the unique object name to the sending object. The parameters determine the initial data of the new object.

'destroy' destroys the specified object.

Communication system

The communication system transports the stimuli issued by objects to the specified target objects. No further properties of the communication system are specified, in particular not with respect to transmission time and the order of transmission of the issued stimuli.

System behavior

The dynamics of a system is determined by the elementary (non-interruptible) steps of behavior triggered when objects receive a stimulus: the data of the receiving object are transformed, and the specified stimuli are issued by the object. Data transformation and stimuli issued are determined by the function and the parameters specified in the stimulus received, and the actual data of the object.

Stimuli may be received from other objects of the system, or from outside the boundary of the system.

Note: this architecture describes systems as having fixed, pre-determined classes. For modeling the transition of a system to a different process model (also called the evolution of enacting process models [CoFF94]) we need to introduce new system behavior by the dynamic creation and destruction of classes. This is beyond the limits of the present paper.

Unique object names

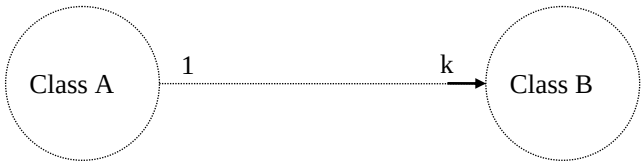
Besides being unique within an object system, object names are disjoint from any other data. There are no operations that may calculate object names from data or from other object names. They may, however, be stored in objects, i.e. be part of the object data.

Object names are used for the identification of the target objets of stimuli, and they may be specified as parameters of stimuli. Note that an object can use an object name only if it was specified as a parameter in its own creation, or if it was received as a parameter in a stimulus.

Object names play the roles of 'access paths' in the system. The knowledge of an object name enables the access to the object. An object whose name is not known anywhere is „dead". The properties of object names are significant for modeling associations between objects and access authorization.

Associations

An object is associated with an other object if it „knows" the name of the other object, i.e. if the name is stored in its data. An object can send stimuli to its associated objects. The association is symmetric if both objects know each other. Associations can be claimed for classes A and B of objects:



with the meaning: each object of class A is associated with k objects of class B.

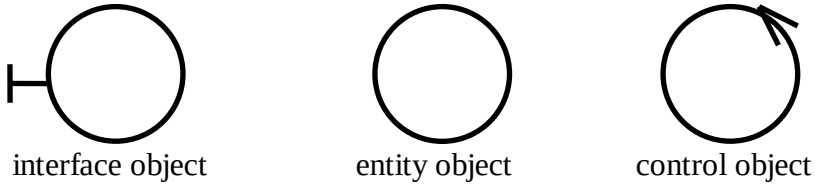
Pragmatic object categories

Following the terminology of Jacobson [Jaco92] we distinguish three types of objects according to the roles they play in a system:

- interface objects are responsible for sending and receiving information, and transforming the presentation of information, across system boundaries, that is for interfacing with other systems and with users
- entity objects are holders of persistent information
- control objects are responsible for controlling processes, i.e. for triggering object behavior between the start and the end of a process.

Note that this is a pragmatic distinction. All these objects are objects in the formal, architectural sense and we do not exclude objects that may not easily fall into one of the three categories.

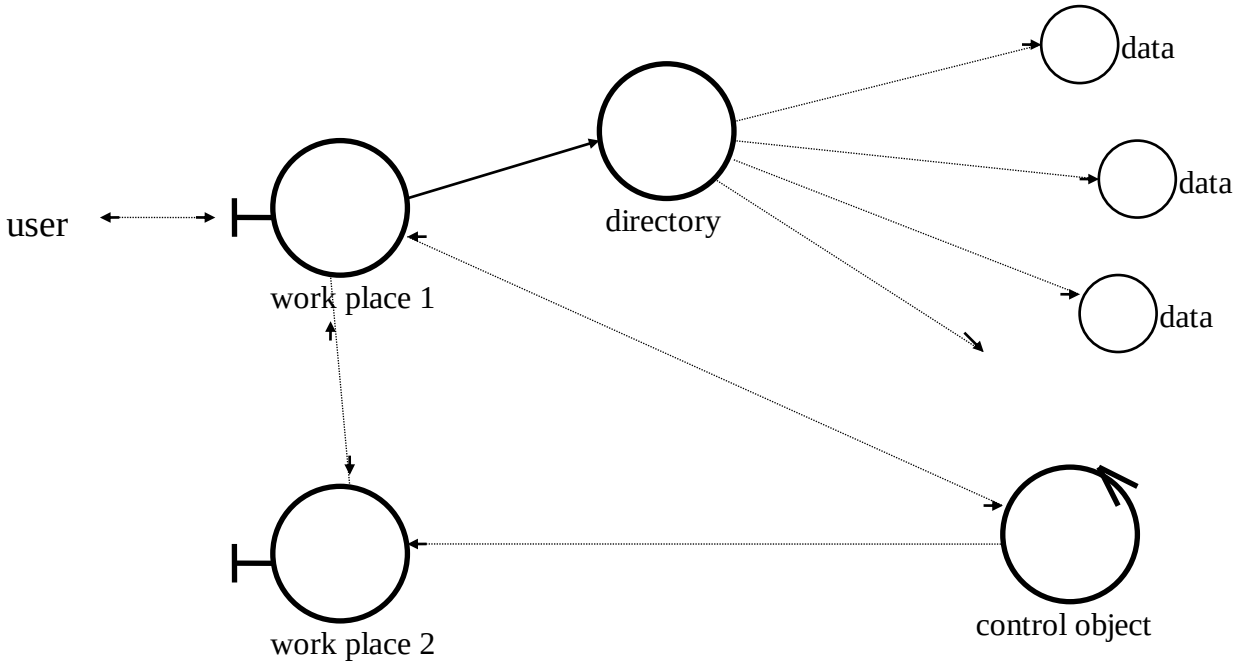
Graphical symbols used are:



General Model of a Work Place

A work place is a system component which interacts with users of the system. It receives user input and displays system output. It is responsible for transforming the presentation of information between internal forms and forms displayable to users. A work place determines the role a user can play. It gives access to relevant information in the system, and it is the interface to the various processes in which the user participates.

A work place is linked to other system components by associations as shown in the following sample picture:



Work place 1 is associated with a directory object, which in turn is associated with data objects. This allows a user of work place 1 to access the directory, sending requests for data identified by an external name, which is translated by the directory into the name of the appropriate data object. A stimulus to the data object then will initiate the return of the requested information to the work place.

Work places may be associated to one another, enabling the exchange of information.

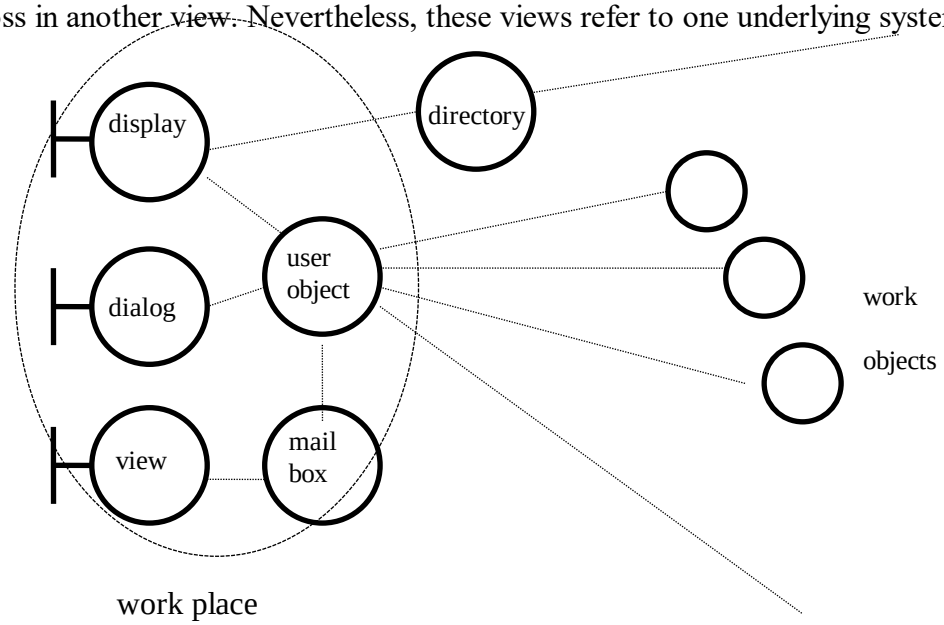
Work places may temporarily be associated with objects controlling a process. Work place 1, for example, may have created the control object and established the association with it. The control object, in turn, may send stimuli to work place 2, if this is to be involved in the process.

We look closer into the structure of a work place object. It may contain:

- interface objects responsible for the receipt of information from the user and the display of information. Examples for these capabilities are dialogs or graphical information displays
- a user object, responsible for holding information about user authorization and for managing user authentication
- associations with actual work objects, and the display of the status of these objects. This signals ‘work to be done’ to the user
- a mail box
- a list of general work requests
- local data files

The total of capabilities of the work place, which means the services available and access authorization to other system components, determines the corresponding user role.

Work places can be linked to many processes at a time. Since communication is asynchronous, there is no problem modeling this situation. It is not necessary to show all the process links in one system structure. For example, we can show the involvement of a work place in a production process in one system view, and the involvement in a communication process with the user’s boss in another view. Nevertheless, these views refer to one underlying system.



work place

Modeling Process Control

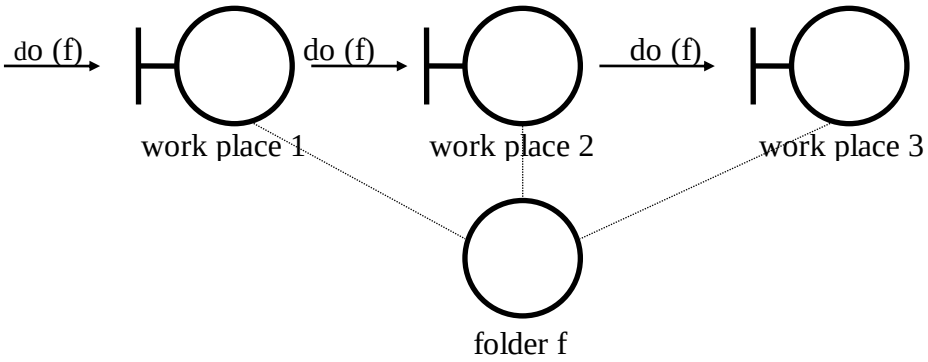
A process is the behavior of the objects of an object system which was started for the purpose of achieving some good result in a particular way. The start event is a stimulus issued either by a person from a work place, or by another object. The running process involves user activities and user decisions through the work place, the creation and destruction of objects, and object behavior triggered by stimuli. Please note that the identity and scope of ‘a process’ derives from the purpose it gets associated with during the analysis phase. System behavior is not formally partitioned into separate processes.

We discuss three different patterns for the control of a process.

Sequential control

A process may consist of steps simply performed in sequence. An example is a folder which needs the attention and work of a series of people. The folder may be handed over from one work place to the next until, for example, final approval is given.

Sequential control is often found in administrative processes. In general it is not advisable to implement level 3 information systems support for it in a straightforward manner. It would mean to hard code it as a property of the work places, or of other objects that are involved. Any change in the process would result in changes of these objects.



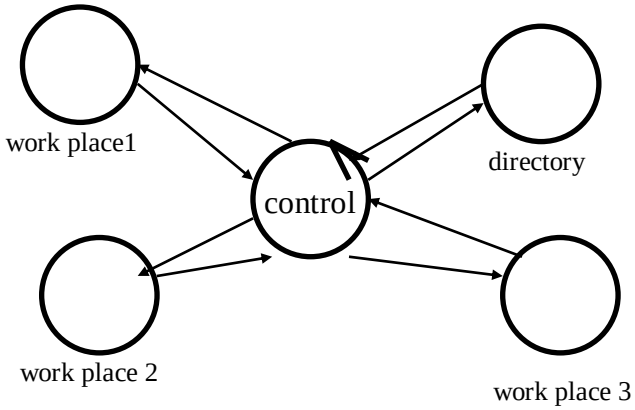
Star-shaped control by a control object

A dedicated control object interacting with work places and other objects may be used to control a process. The control object contains the logic of sequencing, branching and synchronization decisions. It requests services from users by stimuli to work places, and services from other objects. Work places and service objects return stimuli signaling completion of the requested step, or otherwise signaling an exception situation. This leads to a star-shaped structure.

Since a dedicated object is responsible for the process logic, changes in the logic can be handled by changes local to that object, leaving the other objects untouched.

The specification of the process logic in the control object is an essential part of the description of a process. There are various notational systems for the specification of control objects. They must be able to refer to work places and service objects, specify the information sent to and received from these objects, and specify the decision logic determining the object behavior. Implementation of this type of control is the subject of workflow management systems which are able to interpret control descriptions. Control descriptions can resemble activity networks. In turn, activity networks often may be easily mapped on control objects. Their precise meaning is determined by the interaction of the control object with the system [Walk93], [WhFi94], [Walk94].

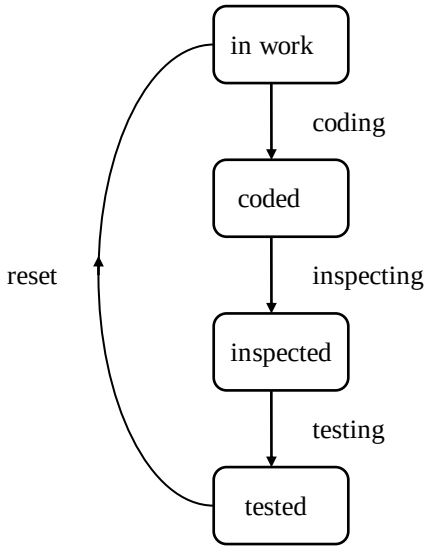
Exercising star-shaped control may also be the responsibility of a user. The corresponding work place in this case figures as the control object.



Control by the status of objects

Process rules can be attached to objects by controlling the applicability of the object functions by the object status. The status is part of the object data and can assume one of a finite number of values. A certain value permits certain functions and locks other functions. The application of a function leads to a new object status. The status rules can be represented by a finite state graph [HuKe89], [ShMe92].

The rules of the earlier example of a code-and-test process are embodied in the following status diagram of a software component:



The status of ‘in work’, ‘coded’, etc. allows the functions of coding, inspecting, etc. These are all human activities, so status setting in this case will not be automatic but must be done manually.

This process construct achieves the following:

- the rules are observed for each individual software component, as we had requested earlier if the rules are inherent in the nature of a work object
- if combined with the construct of a work place, the objects can be associated with the user object as ‘objects in work’, as shown earlier. They represent work to be done by the user. This is called an ‘object oriented work place’
- users have a freedom of choice with respect to planning and selecting work which is offered to them
- work objects can be transferred from one work place to another one without loss of information about the status of work
- in the case of change necessary because of new technical or planning assumptions, or the discovery of grave errors, the status of an object can simply be reset. This then reflects the changed situation, quite independently of where the object currently is attached to, or who has done what to it earlier.

An object status diagram defines the life cycle of an object from creation to an end status, or to destruction, and in fact is the definition of a process fragment (extensively used e.g. in SOCCA [EnGr94]).

In an object system, processes can be interwoven quite independently of which type of control is driving them. It is also possible to go to a next level of detail if this is demanded.

Extending Process Models

Process models can gradually be extended by adding more detail to the description of object properties, by designing the stimuli carrying information between objects, and by adding more objects of concern. Since objects are loosely coupled by asynchronous communication, refinement of different system components may proceed quite independently over longer periods of modeling activity.

Looking at the above code-and-test example, we so far really have introduced only one, rather trivial, rule. There is a longer way to go for adequately defining a test process which satisfies objectives like: delivering repeatable results, permitting statistical evaluation, defining the roles of participating people and defining the required capabilities of tools and libraries. This, in practice, would be achieved by analysis and feedback done by testing and modeling experts and involving representatives of the actual test group. Analysis of how testing is done, or should be done, may result in the following clarifications:

- Testing consists of individual test runs over components with test cases out of a test case library
- The result of a test run is not only tested code, but in particular can also be the discovery of troubles. We introduce trouble reports describing: the test run, the trouble found, and auxiliary attributes like date and time. The trouble may be caused by wrong code, a wrong test case, or even wrong design
- It is not possible to change a component (fixing an error) and running it at the same time. Fixing is done on a copy of the component if fixing and testing are to go on in parallel
- Test runs must be repeated after repair was done, keeping in mind that fixes may be wrong and even might invalidate test runs that were successful before

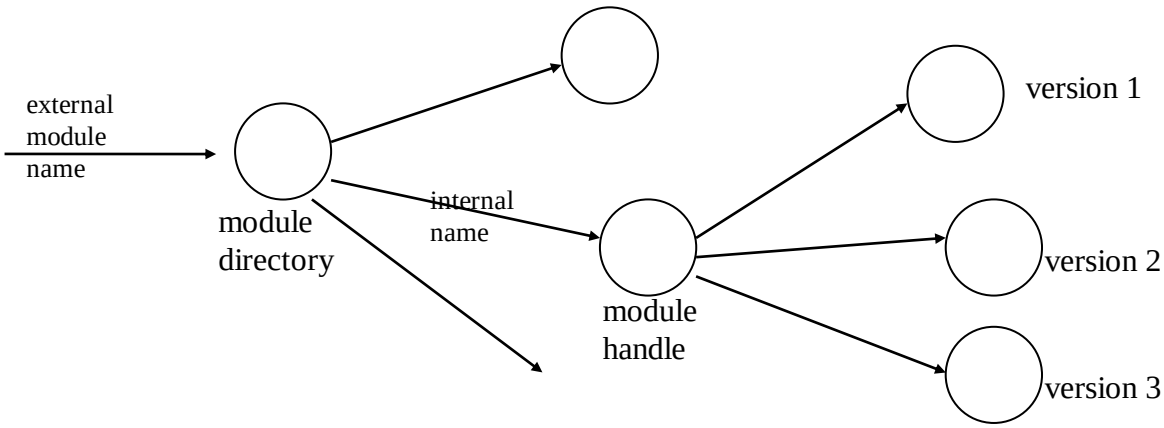
- One does not wish to re-test after each individual fix. A number of fixes will be made in the copy of a component, before this is made subject to further testing. Components, therefore, exist in a number of identifiable versions, which have to be controlled by an appropriate library system.
- Components usually consist of a collection of sub-components (modules) which have been produced by different coders.

All these elements can now be combined in one, consistent model, just using the modeling constructs introduced so far. We would proceed by defining user roles:

- **coder**, having update rights to „owned“ modules, being responsible for interpreting trouble reports, making fixes to code, transferring trouble reports to other work places if they are judged to be the responsibility of another person
- **tester,** having access to components and to test cases, being equipped with an appropriate test environment, being responsible for issuing trouble reports
- **librarian,** having access to a component library, being responsible for building new versions of components.

The main work objects are found to be:

- **components,** consisting of aggregates of modules. Components and modules can exist in different versions. The aggregate arrangement depends on the specific usage requirements and may be the starting point for the design of a library system. For example, the arrangement:



allows accesses: ‘read version 3 of module x’ or ‘read latest version of x’.

Modules are objects, one of their functions being ‘compile’, which creates the corresponding compiled module. Module and corresponding compiled module are associated objects.

- **test cases**

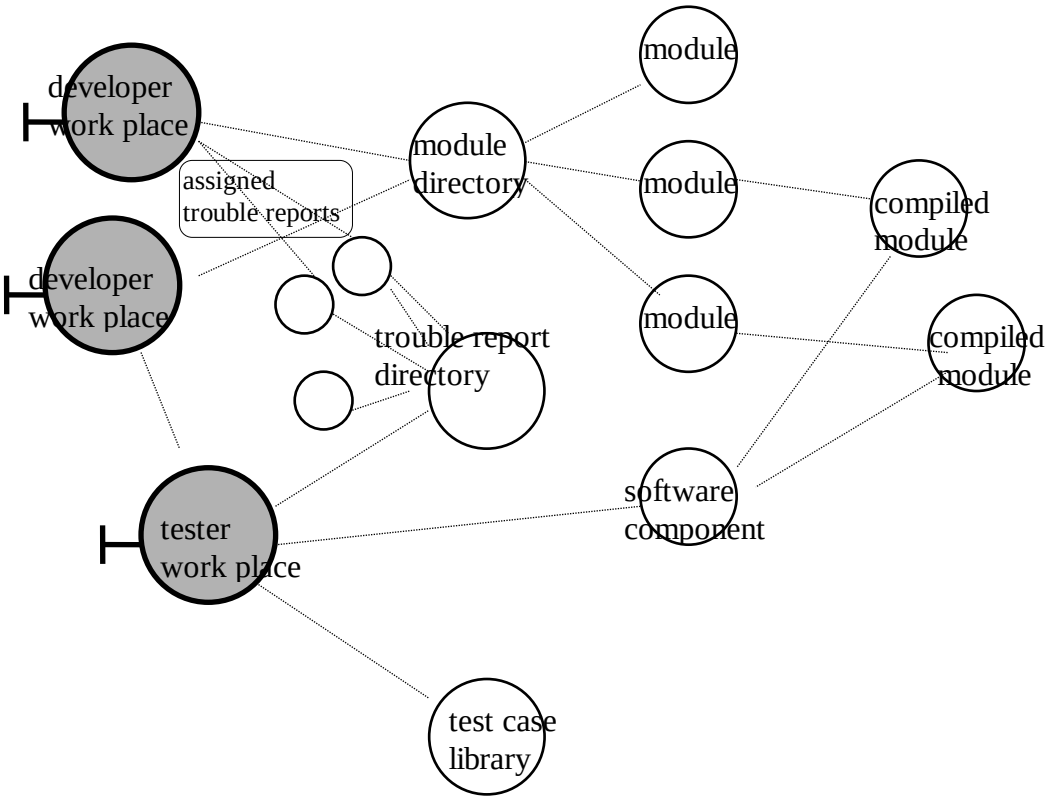
- **trouble reports**, which are characterized by a succession of states like: ‘open’, ‘accepted by a coder’, ‘fix done’, ‘fix tested’, and possibly ‘invalid’. Trouble reports are accessible to all people involved in the process. The role of people determines which states they are authorized to set. Trouble reports

can be sent to specific work places, indicating ‘work to be done’.
The library of trouble reports allows the judgment of the overall process status and can be the basis for process measurements like: number of troubles found in a component, percentage of wrong fixes, number of necessary test cycles.

Work places are introduced according to the capabilities and access rights as demanded by the roles of their users. For example, the work place of a coder would feature:

- read access to modules and problem reports
- update access to modules owned by the coder
- interface objects for the display and update of modules and problem reports
- authorization to set the status of modules to ‘coded’, and the status of problem reports to ‘accepted’ and ‘fix done’
- compiler service for creating object modules
- communication services to other work places
- possibly an interface object for displaying ‘work assigned’ like the identification of a trouble report.

The resulting model is a system of cooperating work places, work objects and aggregated work objects:



The lines between the objects represent associations. The next step then is the design of the type of the associations and the stimuli along these lines. This implies the design of the type of control for the process. Since the trouble reports, for example, represent ‘work to be done’, we may

- either leave it to the users to access the library of trouble reports to find out about work to be done by them
- or create a control object for each trouble report issued, which by interaction with the user work places automatically will carry the trouble through to ‘fix tested’.

The amount and type of automatic control is an important aspect in the analysis.

Evolving a process model can proceed toward the detailed specification of requirements for the various system components, be it user interfaces, dialogs, libraries, tools, or work flow management facilities.

Management and Change of Processes

System components and running processes need to be installed, observed, and changed according to changing needs and out-of-line situations.

It is easy to see that the testing process sketched above may never end: if troubles found lead to fixes which in turn lead to new troubles, the inflation of trouble reports may indicate that the process will not converge. This gives a touch of realism to our process model. On the other hand, we must take precautions to enable the interference with the running process in a controlled way.

We need an additional user role, responsible for observing and, if necessary, changing the process. We introduce the work place of a test coordinator. It has access rights to all information necessary to judge the process status: the trouble reports, the test case library, and all versions of the modules. In addition, test coordination involves the setting of status information on the work objects and influence on the assignment of work. Individual modules may be set to ‘invalid’, requesting re-coding. Authorization of people can be changed, more acting people may be added, test case development can be requested, or the entire process may be stopped and lost effort calculated.

The work place of a test coordinator can be added to the process model developed so far without creating a conflict.

More roles and work places can be added in a similar way. They may concern the responsibility for system management (like performance measurements and creating and destroying system components), managing back-up of vital data, or the management of personnel situations.

Developing Process Models

The development of process models is itself a process, which has to be adapted to the specific needs of an organization. The level of detail to be specified depends on the level of process automation that is aimed at, the available computer system configuration, skills and experience of people, and the size of the organization. There is extensive literature available on object-oriented analysis methods and experience [CoYo90], [WiWW90], [Rumb91], [Jaco92], [ShMe92], [Booc94], [BuCa96]. Most of it, however does not specifically concern itself with the work processes involved in software development. Moreover, there is often a tendency to overload analysis with elaborate notation obscuring the fundamental issues.

Analysis as suggested here will involve the following major steps of identification of system constituents:

Roles of people

It is good to start with describing the different roles of people acting on a process: what is their responsibility, which information flows to and from people, which capabilities are available, what is the interaction with other roles? The basic production roles should come first, coordination and management roles can be added in a second pass.

Acting people are **outside** the system, they just have agents in the system like work places, or other objects representing people, with which they interact. Just note that a system engineer may easily destroy the work place of a person, hopefully not the person.

Major objects

Objects of concern at this point have a certain level of persistency, which means as a rule of thumb: they survive a process.

Associations between objects

Which objects ‘know’ about one another, in the sense of being able to interact with each other by stimuli? How long does the association last?
If an object is addressed only by one specific other object and is passive otherwise, it may be considered just to be data of that other object.

Life cycle of objects

What are the states of objects which differ with respect to the applicability of object functions?

Work places

Work place capabilities need to be analyzed per user role.

Stimuli

What is the information carried by the stimuli between objects? Analysis must determine whether information needed at a certain point is actually available. The fact that an object only 'knows' information it was told at creation, or by stimuli, does make object oriented analysis a powerful thinking discipline!
Part of this consideration is: who creates an object and who destroys it? Note that, at least originally, an object is just known by the object that created it.
Graphical tools for the representation of objects, classes and associations, interaction diagrams and dictionaries for all specification items are helpful for practical modeling. The essence of the approach, however, is the analytical power provided by the underlying system architecture, allowing the modeling of interacting processes and the stepwise addition of detail in a consistent way.

Case Study

The following case study describes a process modeling project which was performed for a middle-sized Austrian company. It aimed at the development of an effective team and process model for small software engineering teams tailoring the ESA software engineering standards [ESA92] to their needs. In this company most development is performed by small and competent teams with multifunctional roles. People must be able to play different roles in one team - designer and developer, sales representative and requirements engineer, project leader and developer of most critical modules, etc. - following the demands of small and multifunctional teams. This is in contrast to the old paradigm of "management by function" in which one person is assigned to one job and does not play any other role. This can only be done in very large companies and even there such a system lacks flexibility. Therefore the paradigm of "management by processes", in which one person can play many roles which are clearly defined within work scenarios, was employed in this project [GIMi92].

During the modeling project

- 9 different scenarios from project acquisition and requirements engineering to maintenance
- 11 roles (and corresponding interface objects)
- 25 data objects
- 40 functions (types of functions were counted, see Fig. A2 - create (W), create (O), create (RD) are e.g. counted as 1 function)

were identified.

The modeling project had a duration of seven months (Feb. 95 - Aug. 95) and comprised

- one consulting team (one expert plus two junior consultants) [BiKM93]
- one field test team (five people working in a multifunctional environment)

who held

- five workshops following the below rules:
  - each workshop had to have an agenda (a minimum set of questions to be discussed)
  - each workshop aimed to include all the personnel involved
  - each workshop resulted in graphical models
  - each workshop resulted in minutes
  - each workshop had a duration of two days
  - each workshop resulted in homework to be done by the consultants and to be reviewed by the field test team.

One of the most critical success factors was motivating the engineers to cooperate with the external consulting team and to use the new guidelines and models. This was due to the fact that managers and software engineers had different viewpoints. The goal of the managers in this project was to gain better insight into the work done by the teams. The goal of the engineers was to improve their work place and to facilitate their work, but they also saw the visibility of the work processes as a means of control that could be used against them by management. Therefore it seems to be of critical importance that modeling is not done with the managers alone. Consultants should include all software engineers and managers in solution oriented discussions identifying roles, responsibilities, work scenarios, and data objects.

Tab. A1 illustrates the steps performed, the main questions that were discussed, and the results achieved. Fig. A1 summarizes some quantitative project data.

The quantitative project data [HaMe93] can show whether the modeling project proceeded properly:

- Number of Inputs from the Workshop

  If key people are left out from the workshop discussions, problems will be detected very late, usually when the engineers 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 realize the requirements and problems the sooner you will get on the right way.

- Cumulative Effort in Weeks and Number of Pages per Man Week

  Let us look at Fig. A1. In the beginning the consulting and the field test team had a lot of discussions and interviews because it really 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. 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). 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 suddenly all engineers and the modeling team start to work towards a shared vision.

Step	**Activity**	**Product**
Planning (Jan. - Feb. 95)	- planning project and allocating resources	purpose, scope
Workshop 1 (Apr. 95)	- identification of different scenarios/use cases - identification of roles involved	
Modeling 1	- documenting the use cases / scenarios - documenting the roles and their interaction	Version 0.1 : describing roles and interface objects and how they interact
Workshop 2 (Apr. 95)	- identification of data objects - scenario diagrams	
Modeling 2	- drawing scenario diagrams	Version 0.2: Version 0.1 plus graphical representation of scenarios
Workshop 3 (May 95)	- refining roles - refining scenario diagrams	
Modeling 3	- refining the role descriptions, the data object descriptions, and the scenario diagrams	Version 0.3: clear and agreed - roles and responsibilities - data objects - work scenarios
Workshop 4 (May 95)	- definition of standard planning templates based on the scenario diagrams - establishment of a data collection and analysis plan for process and product measuremet - discussion of re-use concepts	
Modeling 4	- drawing the MS Project templates - documenting the standard data collection plan - refining the scenario diagrams to include data collection, analysis - including the aspect of re-use in the scenario diagrams	Version 0.4: Version 0.3 including - standard planning templates - data collection and analysis plan
Workshop 5 (July 95)	- review of the overall model - acceptance test protocol	
Modeling 5	- final refinements	Version 1.0
Acceptance (Aug. 95)	Final Acceptance	Version 1.0 Released

Tab. A1: General Project Schedule

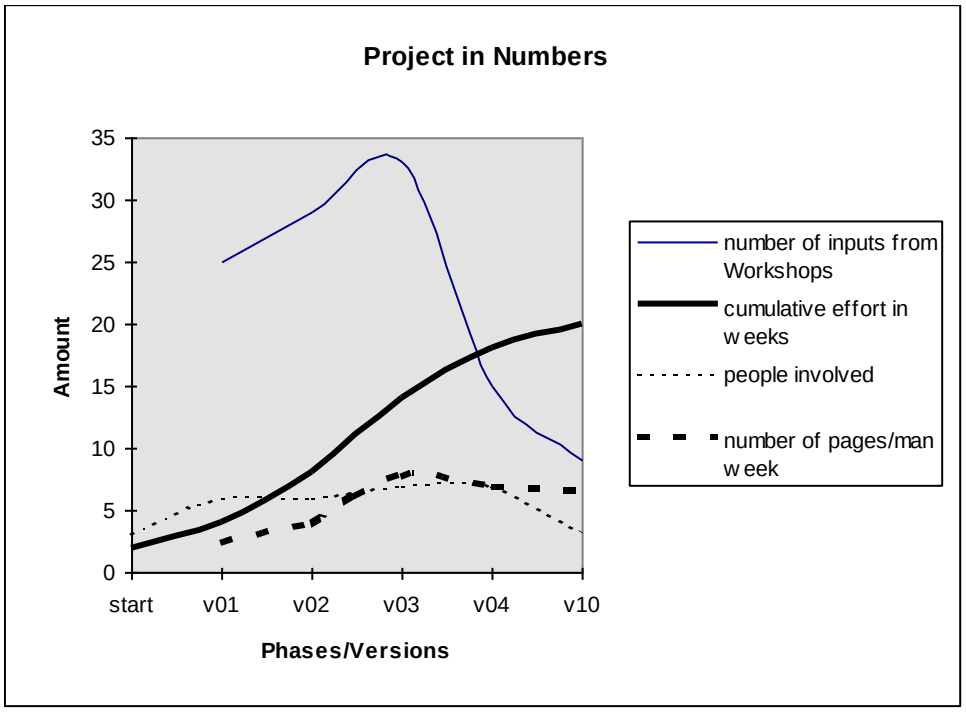
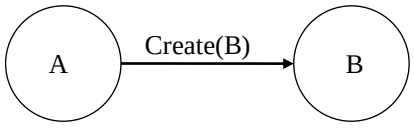


Fig. A1: Quantitative Data of the Above Modeling Project

Figs. A2 and A3 show a sample part of the project acquisition work scenario and the underlying data object associations. The work scenario shows the interactions between the work places, and the interactions between work places and the work objects. The work places are labeled by the corresponding roles of the acting people. Note that the work place of the division head plays the role of a control object!

Functions 1 to 13 might be repeated many times until the customer sees all his wishes satisfied in the offer. Then functions 14 to 17 apply and a project leader is appointed to start the software development process.

As a shorthand, the following graphical notation is used for describing the creation of objects:



with the meaning that object A creates an object with the name B.

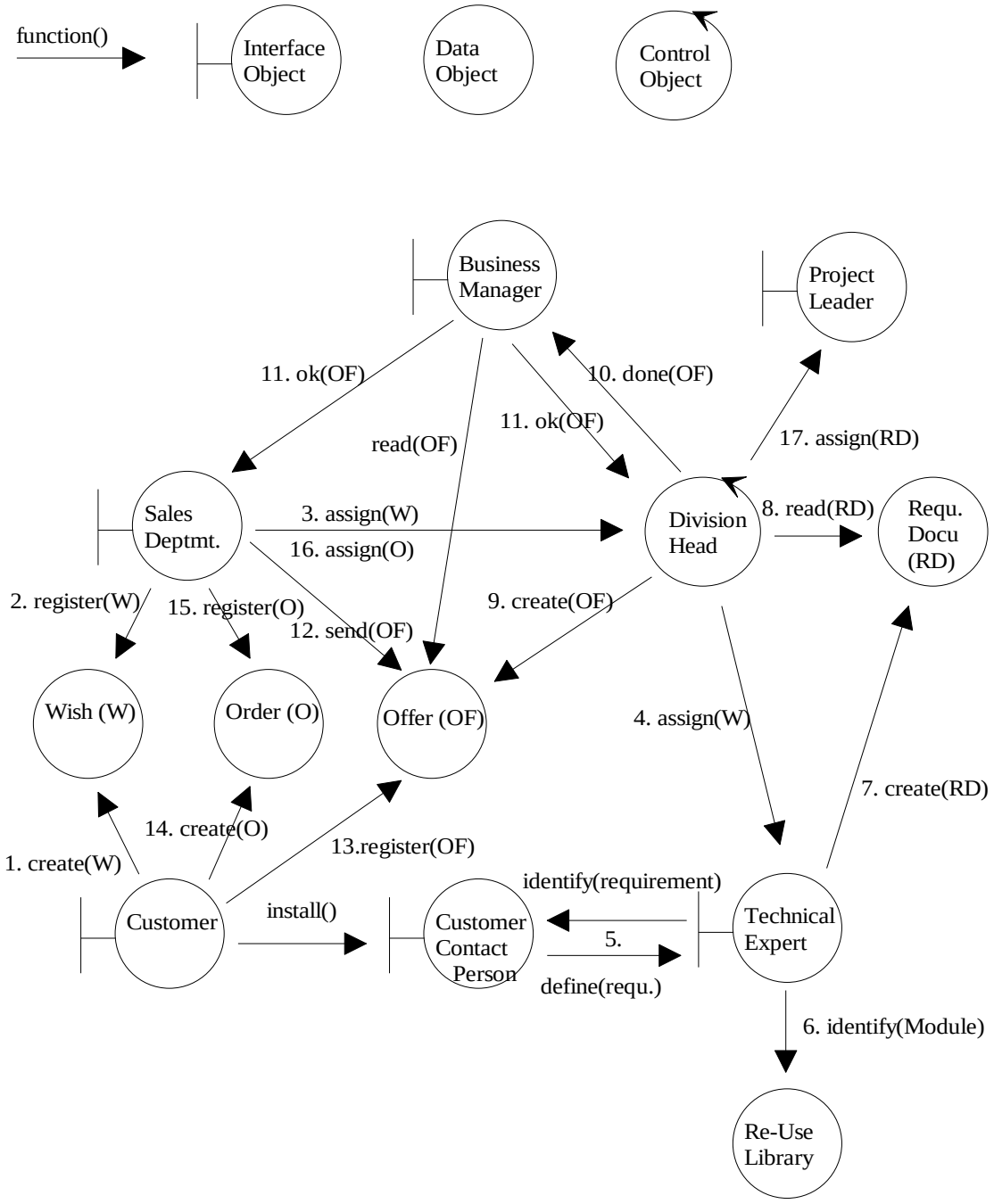


Fig. A2: Parts of the Project Acquisition Process

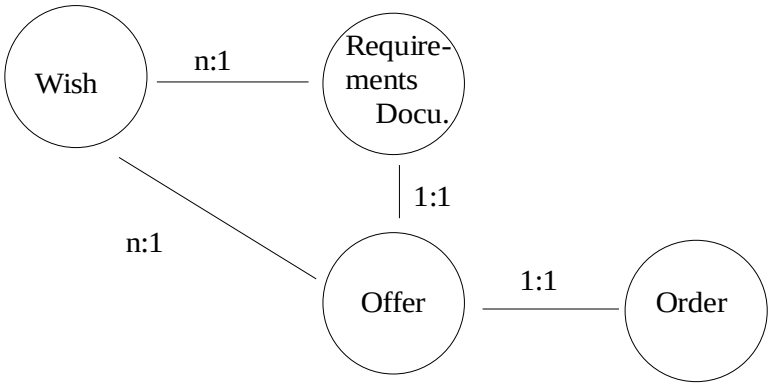


Fig. A3: Underlying Data Associations of the Process in Fig. A2

The team benefited from the results of the modeling exercise, which were:

- visible and understandable roles
- visible and understandable work scenarios
- a better interface control in the team
- a multifunctional team structure (e.g. three people playing nine of eleven roles)
- a better understanding of process requirements as defined by e.g. ISO [MeKu94]

Most modeling approaches start with the identification of process steps, connect them to a network of activities (a workflow), and then assign roles to the activities. However, in this modeling project the human resources were regarded as active parts of a system who can largely contribute to the successful implementation of a work scenario. Also it was much easier to motivate people to think about the work environment if they feel personally involved in the process. Thus the project started with a set of interviews identifying the roles, the communication between the roles, the objects and results produced and exchanged by the roles, and the people playing the different roles. This way of thinking (role based modeling instead of activity based modeling) is much more supported by the modeling technique presented in this paper than by the usual activity driven charts. This way a project becomes a role-based model and to initiate a process a group of people is assigned to the roles.
Within each modeling step there were three learning stages. First we identified the current state and established a model visualising the current roles, responsibilities, and workflows. Second the model was presented to all the staff and they started to discuss improvements under guidance of an external expert. So, for instance, the support people were complaining that so far no acceptance test protocol plus all unit and system test protocols were delivered to the support after acceptance, so that the re-test was always difficult (always they had to re-invent all test cases instead of using existing test protocols for regression testing). Third the model was refined, presented again to all staff, and if accepted the model became part of the overall development guideline.
The customer did not accept a workflow management tool because (i) organisational aspects were emphasised more than the introduction of a new workflow technology (if you do not know the model you cannot use a workflow tool), and (ii) a workflow management tool would have restricted the freedom of developers who sometimes could make different steps (and the team wanted to use the model as a guideline but allow creativity and flexibility). In addition to that they used a SA-based tool for design which already had become a company culture (everything was discussed in SA charts) so that it was decided to draw everything with the design tools available in the software division.

References

[ArKe94] Armitage J.W., Kellner M.I., A Conceptual Schema for Process Definitions and Models, in: *Proceedings of the Third International Conference on the Software Process,* October 1994

[BiKM93] Biro M., Kugler H-J., Messnarz R., et. al., BOOTSTRAP and ISCN - A Current Look at a European Software Quality Network, in: *The Challenge of Networking, Proceedings of the CON´93 Conference*, Oldenbourg, 1993

[Booc94] Booch G., Object-Oriented Design. Benjamin/Cummings, Second Edition, 1994

[BuCa96] Buhr R.J.A., Casselman R.S., Use Case Maps for Object-Oriented Systems. Prentice-Hall, 1996

[Chro92] Chroust G., Modelle der Software-Entwicklung. Oldenbourg, 1992

[CoYo90] Coad P., Yourdan E., Object-Oriented Analysis. Prentice-Hall, 1990

[CoFF94] Conradi R., Fernstrom C., Fuggetta A., Concepts for Evolving Software Processes, in [FiKN94], pp 9-31

[CuKO92] Curtis B., Kellner M., Over J., Process Modeling, CACM 35, 9 (1992), pp. 75-90

[EnGr94] Engels G., Groenewegen L., SOCCA: Specifications of Coordinated and Cooperative Activities, in: [FiKN94], pp. 71-100

[ESA92] ESA Board for Standardisation and Control, ESA PSS 05 Software Engineering Standards, Paris, France, 1992

[FiKN94] Finkelstein A., Kramer J., Nuseibeh B. (Eds.), Software Process Technology, John Wiley & Sons, 1994

[GIMi92] German Interior Ministry, German V-Model, Bonn, August 1992

[HaMe93] Haase V., Messnarz R., et.al., Process and Product Measurement, IIG Report Series, Report 368, Institutes for Information Processing Graz, June 1993

[ISPW-6] Kellner m., Feiler P., Finkelstein A., Katayama T., Osterweil L., Panedo M., Rombach H.D., ISPW-6 Software Process Example, in: *Proceedings of the First International Conference on the Software Process,* Washington 1991, IEEE Computer Society Press.

[Hump89] Humphrey W.E., Managing the Software Process. Addison-Wesley, 1989

[HuKe89] Hunphrey W.E., Kellner M.I., Software Process Modeling: Principles of Entity PROCESS Models. *Proceedings of the 11 th International Conference on Software Engineering*, IEEE, 1989, pp. 331-342

[Jaco92] Jacobson I., Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1992

[MeKu94] 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

[Rumb91] Rumbaugh J. et al., Object-Oriented Modeling and Design. Prentice Hall, 1991

[ScPo95] Schaefer W., Botella P. (Eds.), Software Engineering, *5th European Software Engineering Conference,* Feb. 1994, Springer LNCS 989

[ShMe92] Shlaer S., Mellor S.J., Object Lifecycles Modeling the World in States. Prentice Hall, 1992

[Warb94] Warboys B.C. (Ed.), Software Process Technology, *Third European Workshop,* Feb. 94, Springer LNCS 772

[Walk93] Walk K., Workflow Management Concepts, A White Paper. IBM Vienna Software Development Laboratory, November 1993

[Walk94] Walk K., Object-Oriented Development of Workflow Management Applications. A Development Method. IBM Vienna Software Development Laboratory, July 1994

[WhFi94] White T.E., Fischer L., Eds., The Workflow Paradigm. Future Strategies Inc., Alameds, CA, 1994

[WiWW90] Wirfs-Brock R., Wilkerson B., Wiener L., Designing Object-Oriented Software. Prentice-Hall, 1990

The Place of Formal Methods in Process Improvement

Dr. Andrew Butterfield
Dr. MÌche·l Mac an Airchinnigh[29]
Department of Computer Science
University of Dublin
Trinity College, Dublin 2, Ireland

ABSTRACT

This paper discusses the application of formal methods technology in the software development process in industry. We describe formal methods and identify those parts of the development process to which they are best suited. We then discuss various industrial experiences with formal methods, with a strong emphasis on how those techniques helped (a) to formulate third-party tenders and their responses and (b) to identify and resolve serious errors and misconceptions. Finally we look at the practical issues involved in introducing formal methods into an industrial environment. Attention is paid to the differing aspects of both education and training with a particular focus on the psychology of the organization, human factors development and management, and the timing of new technology injection. Throughout we discuss some myths and misconceptions that are prevalent concerning the nature and efficacy of these techniques.

INTRODUCTION

The last ten or fifteeen years has seen the emergence of a new technology, which is commonly known as Formal Methods, which purports to provide a considerable improvement in the quality of software development. The claim is based on the reliance on mathematical formalisms and proof, leading to the use of terms such as „proof of correctness“, or „derivation of programs by correct transformation“. However, the technology has not been adopted on a wide scale, and is commonly perceived as being unwieldy and unworkable in practice, with too many claims being based on small or ‘toy’ case studies.

This paper seeks to clarify the issues surrounding this new technology, and to explain the circumstances for which it is best suited. We place considerable emphasis on the need to be very careful when introducing a new software development or quality assurance technology, such as formal methods, in order to ensure that it fits in well with existing processes and practice in the organization in which it is being introduced.
The discussion is backed up by reference to the industrial experience of the authors in applying and introducing formal methods to large software and systems developers, particularly in the telecommunications field.

WHAT ARE FORMAL METHODS?

Formal Methods are mathematical techniques for describing and analysing computing systems. In general a formal method can be viewed as having the following main aspects:

1. Notation---a formal method will have a well-defined notation or language used to describe models of software components and other computing systems

[29] This work was supported by K&M Technologies Limited, of which the authors are consultants.

2. Proof System---given descriptions in the notation, we need rules which both explain the meaning of the descriptions, and how to reason about the systems so described. It is this aspect of formal methods which involves the often cited notion of proof.

3. Method---as the name „formal method“ implies, there must be a method---a series of guidelines on how to use the notation and proof system to achieve various system design goals, such as specification, system decomposition, verification or validation. Not surprisingly, it is in this aspect of method that we find the strongest links with the concept of a software development process.

4. Tools--- while many small formal descriptions can be written and analysed by hand, in practice we need tool support to assist the formal methods user. An obvious role for tools is to support the reasoning process by helping to apply rules or check their application (so called proof assistants or provers). Another key role for tools is to ensure the correct application of the method---in this area such tools are similar to those produced to support workflow processes.

It is worth pointing out that, particularly in the U.S., that the term „formal methods“ is often taken to mean any process that uses a tightly defined notation and series of development steps (e.g. JSD), often with no proof aspect at all. However, in this paper, we use the term to refer solely to techniques which have a strong mathematical underpinning, in all three aspects---notation, method and proof.

Benefits of Formal Methods

Formal Methods bring rigour to the process of software development. By their very nature, they encourage users to perform an extensive and thorough examination of the system under study.

For example, very often, a formal method, requires a user to perform a proof, of a certain (well-chosen) property of some system description written in its notation.
This is an example of the interlinking of notation, proof and method which has an intended side-effect of requiring the user to check all relevant aspects of the system under study. It is not possible to produce a rigorous proof by skipping over ‘obvious’ parts---this is often where serious errors lie, and is a good motivation for having a method with tool support.

Again, many formal methods have the notion of an „invariant“, which is a precise descrpition of properties of a system that should always hold. There is clear evidence (Jones 1996) that the stipulation in a formal methods requiring such invariants, leads to a clear and very useful record of the safe assumptions regarding the system's state. This proves very useful to designers and coders, and is also a useful archive resource for subsequent maintenance effort and future plans to modify and enhance the system.

Case studies have also shown (Brookes, Fitzgerald and Larsen 1996) that teams applying formal techniques tend to ask more questions of the „customers“ regarding the requirements and specifications of the required system, and will often raise issues that were overlooked by the customers, but of real concern to them.

Formal Methods and the Software Process

The key benefit touted for formal methods is its ability to prevent or intercept errors in the system and software design lifecycle. Given that there is a cost associated with using a formal method, as distinct from an informal one already in use and, therefore, already budgeted for, it seems reasonable to assume that formal methods will be most cost-effective if they are used in the software process at points where preventing or detecting errors pays the highest dividends. In particular it is worth keeping in mind that the most expensive errors are those introduced at an early stage of the process, but not discovered until late.

Myth: It is a commonly held view that a development using formal methods is more costly (labour and time) than one using informal methods.
While this is often true, there are an increasing number of case studies (Formal Methods Europe WWW site at http://www.tcd.ie/FME/) showing that the use of formal methods often reduces the development time or labour cost or both.

Myth: It is also often stated that the increased cost is not justified by the resulting quality increase.
If you are a mass-market software developer, producing word-processing software for instance, this is almost certainly true (time to market and low pricing are paramount, while your customers will tolerate bugs, and you can also increase revenue by selling fixes). However, if you are writing safety critical software, this is most definitely not the case---indeed formal methods are already used to an increasing extent in this market sector. Another interesting example is a consumer electronics company
using formal methods to assure the quality of embedded software, because it cannot afford to ship faulty (buggy) electronics products to its customers (they are buying an electronic gadget, not software, and will rightly expect a refund if the gadget does not work). It is interesting to speculate on the popularity of formal methods among software product developers if users could get money back for, or sue, for the consequences of software bugs.}

Given all the considerations above, it is not surprising to find that the main areas where formal methods have been applied cost-effectively are in the early phases
of the process life-cycle (Requirements Capture, Requirements Analysis, System Specification), where the system description is still relatively small (thus reducing the volume of formal description required) and the cost of an un-detected error is very high (resulting in the use of careful and rigorous techniques early on being time well spent).

In principle, formal techniques can also be used during the design and coding phases (proving programs correct). However, the cost increases as the complexity of the design description gets larger and closer to implementation level, while the cost of errors in the later life-cycle phases are relatively low. In practice, the use of formal methods at the coding level requires considerable effort and powerful tool support, for all but the smallest projects.

Interestingly, there are a lot of benefits to be gained by using formal methods to link the specification and test phases (Paleska 1996). Testers need a reference against which to test, a reference which is usually a specification at the appropriate level for the type of test. Formal techniques are proving very useful in assisting test suite designers by providing quality test material.

Experiences with Formal Methods

Through K&M Technologies Limited the authors worked on various projects in industry, applying formal methods to various parts of the development process lifecycle. We discuss these experiences here, with particular emphasis on the benefits that arose from the use of formal methods.

Each experience is introduced with a description of a scenario which sets the scence for the application of the Formal Method. In order to avoid the technical details usually associated with specific problem domains which rend to cloud the key issues, we have chosen to recast the problems in terms more readily accessible to all. This has the added benefit of protecting the identity of the client/customer in each case which was a large multi-national corporation. In addition to understand some of the key results, it is important to note that in each case the client/customer organization was comprised of large traditional engineering groups/divisions in addition to the ‘more recent’ software (engineering) group/division.

Experience 1

Scenario: Consider a large public library. It is organized in the traditional manner around the basic concepts of book, catalogue, reader, librarian, etc. It is required to 'computerize' the library in the sense that basic concepts shall be open to fresh interpretations/implementations. For example, the 'book' may be a piece of audio, or film, or electronic text, or any combinations of these. In fact it may even be a book in the traditional sense. The 'reader' may be human or a machine. The concepts of catalogues, borrower, etc., must all be revisited in the light of the computerization.

Practice: Two different models of a portion of the same system were constructed. Each model was effectively a different view of the system. One corresponded to the traditional view of the library. The other corresponded to the computerized view. Then an invariant or intrinsic property of the system was identified. For example, in the case of the library, if a reader has a book then this fact is recorded. When one attempted to verify the property, established for the traditional model, with respect to the computerized model, a serious inconsistency was determined.

Lesson: Do not rely on one model alone. Products of sufficient complexity give rise to difference views. Checking for cross-consistency will usually identify errors in understanding of the requirements.

Experience 2

Scenario: Consider the corporation which has built up a large subscriber base for its products and automated it. Due to historical circumstances, much of the technical intricacies of the in-house computing system are distributed between one engineer's memory and some incomplete documentation. The corporation has expanded to such an extent that it must tender for a new hardware/software database platform appropriate to its telecommunications needs and rigorous fault-tolerance requirements.

Practice: The tender document had to be a succinct and precise English text giving all of the necessary requirements. The engineer responsible was debriefed over an intense two day period and a formal model constructed within one week. A draft text was constructed and analysed. The final text delivered to the suppliers was backed up by a formal model. Although there was no requirement that the delivered product conformed to the formal model, there was an earnestness on the part of the suppliers to be seen to do as well as those who had prepared the requirements text.

Lesson: A formal model clarifies the real requirements and may be used as a guide in constructing the requirements document. All concerned take the matter with a great deal of earnestness and have greater confidence in the written text, knowing that there is a model behind it.

Experience 3

Scenario: A particular product had to be developped by three distinct groups, which we shall denote by A, B, and C. We will use the same letters to designate the components built by each group. Such was the nature of the product that information was gathered by C and passed on to B. The component at B processed some of the information and passed on a summary to A. In addition any information from C not processed by B was also passed onto A. A was responsible for processing all information received from B and controlling the behaviour of the overall product, at least as far as the end-user was concerned. Each group had its own set of requirements.

Practice: The construction of a formal model arises necessarily in an attempt to understand the (functional) requirements. In the course of the consultancy, it was determined quite quickly that there was a serious problem at the interfaces between the groups. In particular, 'junk information' was being passed through B from C to A and A was proposing to store it. The result would have been disasterous for the product as a whole.

Lesson: The solution to be problem was obvious. There must be a single master requirements document and, consequently, overview model. Often, in the course of modelling, information has to be sought from various groups. Such an activity may lead to an identification of management problems or problems in the overall process.

Experience 4

Scenario: A corporation needed to get a product out into the marketplace *in time* order to remain competitive. Management believed in the usefulness of semi-formal design methods with the appropriate tool support. A team of around fifteen was assembled and given the usual sort of industrial training. The project was launched and the design commenced. In addition, a state-of-the-art tool which supported the method in question was acquired at considerable cost.

Practice: During the consultancy two facts emerged. First, in spite of the hard work and enthusiasm of the team, the recommendations of the method were not followed. Only a single iteration of the design had been attempted and the design had been processed by the tool with approximately 500-plus pages of pictures and documentations. All the relevant information was recorded in a database maintained by the tool and printed out as appendices. The initial reaction was that the team had failed singularly under pressure of deadlines. However, there appeared to be another reason. Upon interviewing the team members, it emerged that any change made to the design would have rendered the database inconsistent. In other words, the tool was seriously flawed.

Lesson: The project was ultimately abandoned at a considerable cost running into tens of millions of ecus. There were other reasons apart from the design failure. Even if one does not set out to build a formal model or to use formal methods, the mind-set of the formal methodist can be utilized to great effect.

Summary of Experiences

In the case of the perceived level of difficulty of the mathematics underlying the Formal Method used, the best remark is that which we quote here from a (software) engineer in one of the customer organizations who exclaimed most enthusiastically

„It's not rocket science math!"

The mathematics is certainly well below that expected of any (ordinary) engineer in an organization.

We never attempted to inject formal methods into the mainstream process within any organization. It was evident that the use of formal methods had to be complementary to whatever process was in place. Our experience in the field bore this out. We do recognize that there may be certain types of product or product component where it would be essential to have inline formal methods. But, in general, the formal methods activity will be alongside and apart.

Finally, it is worth noting that computer technology has an enhancing effect on process. If the process is bad then the computer technology will makethings worse. Formal methods have a real role to play in identifying and exhibiting faults in the process.

Adopting Formal Methods

Given that an effective software development process is in place, we now turn to the issue of how best to introduce formal techniques. We consider that a succesful introduction of formal methods requires the following things:

1. Champions
2. Critical Mass

3. Gradual Integration
4. Education
5. Time

We shall now elaborate upon each of these items.

Champions

As with any new technology, there will often be considerable natural resistance to the introduction of formal methods. To overcome this inertia, it is important that the adoption of such techniques be overseen by a „Champion"---a highly motivated individual who understands the advantages of the new technology, and is willing to maintain forward momentum in the introduction of the new concepts and ideas that come with this new technology. This observation based on the authors' own experience, has been made by others well placed in the software engineering
field (McDermid 1996). In the language of Formal Methods, the champion becomes the model owner on behalf of his/her organization. It is essentially in this way that effective technology transfer in formal methods takes place with the organization.

Critical Mass

It is important, especially in the early stages of adopting formal methods, that a sufficient number of people in the organization are involved in its introduction to ensure that a form of critical mass is reached. As a rule of thumb, in the psychological process of technology transfer, we estimate the size of the critical mass to be five to seven people. If fewer than about five people are involved, the whole venture runs the risk of being stagnated, with the small group becoming increasingly isolated. It is important that there are enough people involved to keep the ideas
flowing, and to share the task of integrating the new technology with the work practices of their colleagues

Gradual Integration

The adoption of formal methods requires a certain change in the kinds of concepts used, and the culture surrounding, the capture and analysis of requirements, and the production of specifications. Like all conceptual and cultural changes, a gradual approach to their introduction is much more likely to succeed, than a sudden switch. In particular, as far as process development is concerned, it is very important to ensure that the adoption of a formal method does not lead to a severe or abrupt change in working practices at a critical point in the life cycle.

Many attempts to introduce formal methods get into difficulties because the technique is inserted in a 'cut-and-insert' manner to an existing process (see Figure 1).

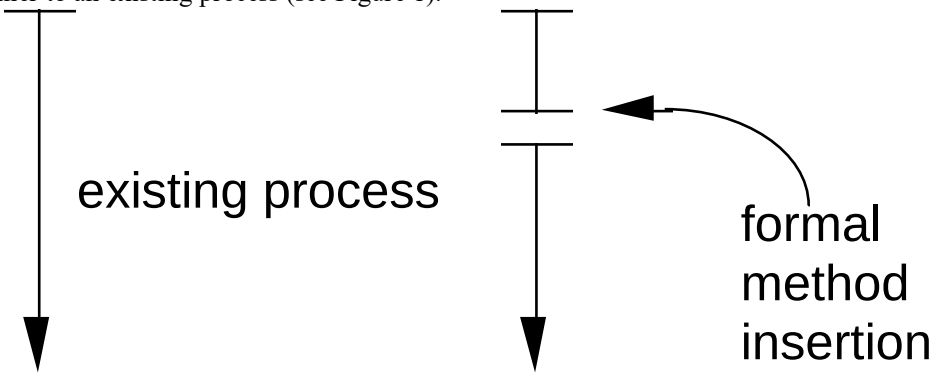


Figure 1. Cut-and-insert

This results in a considerable disruption and delay in the development life-cycle, which increases the stresses and pressures on the project, as deadlines loom up while apparently little progress is being made. An often more effective approach is outlined below.

A key issue to be addressed in real software development environments is that often a new project involves making modifications to an existing system, rather than starting the design and implementation of a system from scratch. A problem with this is the difficulty of using formal techniques to describe a small set of „delta-requirements“,
when the vast bulk of the system to be modifed has been previously described in an informal manner. While formal techniques can be effective here, giving a precise picture of the new requirements, we can see a clear benefit in using such techniques to archive precise descriptions of what has been built, as well as what is about to be built. A formal specification or model should be viewed as both a prescription of what to build, and a archive of what was done, to be exploited when the next modification cycle is begun.

One approach to the introduction of formal methods is to exploit this notion of archiving. Rather than disrupting a new development by the cut-and-insert method of formal methods introduction described earlier, we propose the adopt ion of a parallel ‘develop-and-record’ approach instead.

During the first product development in which formal methods are to be introduced, we simply run formal analysis and modelling in parallel with the pre-existing process. This minimises the disruption and ensures that the project continues to make progress. Meanwhile, those learning and applying the techniques (usually on part-time basis) get a feel for how they relate to the production process, and get some oppurtunity to give feedback to that process when they uncover errors and misconceptions. Apart from the increase in knowledge about formal techniques, and the skills to apply them,
a significant gain is made in that a formal description of an existing product becomes available.

Subsequent changes can then exploit the fact that formal descriptions of the system to be changed are already in place, as are the personnel who developed them. The results of the previous development can then feed into the subsequent developments, and the role of formal techniques can be broadened and deepened, drawing on the earlier experience (see Figure 2).

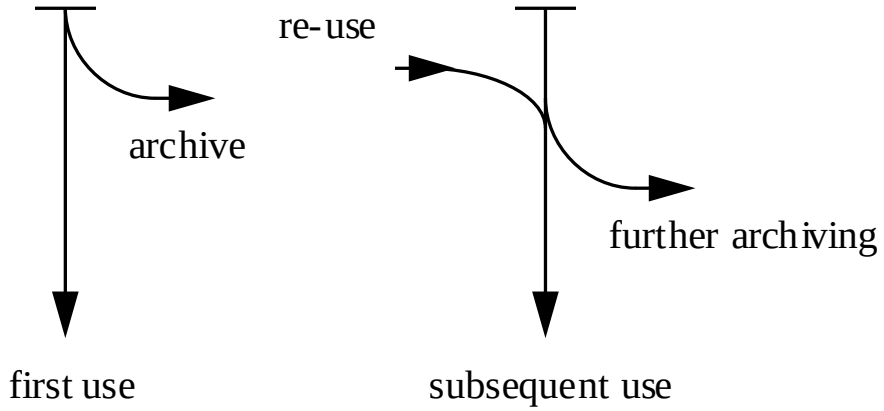


Figure 2. Develop-and-Record

This approach of gradually bringing in formal methods over several product development lifecycles has the advantages of minimal disruption to existing processes. It also ensures that a library of formal descriptions of existing products are built up over time, considerably reducing the cost of subsequent developments. Moreover it is clearly appropriate to product families.

It should be stressed that the archival benefit of formal models is lost, just as with informal descriptions, if the process does not take care to ensure that any design or coding changes are reflected back into the requirements and specification documents. This just reiterates the fact that no formal technology can overcome a deficient development process.

Education

As with any new technology, the intended users will have to be educated in its principles, and trained to apply it and its tools effectively. As formal methods is now an integral part of many third-level curricula, the education step is at least easier today than it was when formal methods first started to appear in industrial settings. Extensive training in the application of various methods is also available.
But it should be kept in mind that such training will invariably have to be tailored to meet in-house process requirements.

Formal methods does not require „rocket science“ mathematics, despite the large number of highly technical and erudite papers published on the subject each year. A clear distinction needs to be made between the „behind the scenes“ (a.k.a „foundational“) mathematics used to establish the soundness of the methods, and the application-oriented mathematics used by the working practioner. Many academic papers on formal methods are biased towards the foundations, and only use applications as small examples to illustrate these deeper issues. The distinction is somewhat akin to that between a programmer and a compiler writer. Programmers using a language such as Pascal, C, C++, Ada, Modula-2, Java, do not need to know how to write the compilers or interpreters for them. Instead they only need to be able to write in those languages. On the other hand there are occasions, rare though they might be, when chief programmers do need to know the limitations of their languages and will be expected to understand the technologies underlying them.

In a similar vein, it is worth remebering that far more people involved in a software process are consumers of specifications than are producers. In other words, the team that writes the specifications is smaller by at least one order of magnitude than the design or coding teams that refer to them. The consequence is simply that it is not necessary for everyone who comes into contact with formal specifications to be able to write and develop them in an effective manner. The ability to read and understand is important. But this is an easier skill to acquire.

Time

Formal methods is not a plug-and play technology, and requires time to adopt properly. The key to its succesful adoption is its gradual adoption over time---bit by bit. Typical lead times from the date of initial introduction to that of significant payback (as determined by profits/quailty improvement of the end-product in the marketplace) may be of the order of a few years for a large project, though some studies do show much quicker returns (Brookes, Fitzgerald and Larsen, 1996).

It is important to remember that a little formal methods is better than none, and that any increase in their use brings an increase in benefits, often at very little or no extra cost. There is no need to adopt formal methods in a large-scale manner (jumping in at the deep end), an approach which often leads to frustration and disillusionment.

The key to succesful adoption of any new technology is: Appreciation, Use, and Exploitation. First introduce it on a small scale so it benefits can be appreciated. This appreciation phase must embrace management at all levels. Secondly acquire the skills to be effective by extending its use to a range of projects. Finally reap the dividends as one becomes more experienced in the effective application of the technology. This is the exploitation phase where mastery of the technology beats the competition.

Where do I find out more?

The European Commission sponsors a committee called Formal Methods Europe (FME) whose role is to assist in the transfer of formal methods technology from research environments into industrial practice. The FME Committee is also responsible for running a conference on formal methods, with a strong emphasis on industrial experience and case studies. The committee can be contacted through its secretary, currently:

Formal Methods Europe,
c/o John Fitzgerald,
Centre for Software Reliability,
Bedson Building,
University of Newcastle upon Tyne,
NE1 7RU, UK

One emerging resource of information on formal methods is the World Wide Web (WWW). Of particular note are the web-sites being developed by the FME committee, under the aegis of ESSI dissemination actions FMEInfRes (21375) and FMEGuides (21703), as well as a series of seminars planned under the FMEIndSem project (21377) for 1997. The official FME homepage is at „http://www.cs.tcd.ie/FME/“ and contains links to a wide variety of resources, including databases recording information about industrial applications of formal methods, as well as details of the various methods and tools available. Another very worthy site is Jonathan Bowen's Formal Methods virtual library at „http://www.comlab.ox.ac.uk/archive/formal-methods.html“, as well as a book co-edited by him with Michael Hinchey (1996) which gives many descriptions of industrial experiences.

Conclusions

We wish to acknowledge the assistance of all those anonymous engineers who met the challenge of formal methods in their organizations and took them in their stride. Thanks are also due to our colleagues in Formal Methods Europe who endured our opinions over the years. A specific note of thanks is due to Professor John McDermid of the University of York, England, who made available to us in September 1996 his most recent findings in Software Engineering.

In conclusion, we can offer the following recommendations regarding the introduction of formal methods into a software development process:

1. Have a proper process in place
2. Find a champion willing to run with it and ensure they have enough resources to reach a critical mass
3. Remember---formal methods are developed to help your process work more effectively, so integrate then into your process, rather than disrupt your process to suit them.

References

Bowen J. and Hinchey M. (eds). 1995. Applications of Formal Methods, Prentice Hall International Series in Computer Science, ISBN 0-13-366949-1.

Brookes T. M., Fitzgerald J. and Larsen P.G. 1996. Formal and Informal Specifications of a Secure System Component: Final Results in a Comparative Study.
In Gaudel M.C. and Woodcock J. (eds). 1996. FME'96: Industrial Benefit and Advances in Formal Methods, *Lecture Notes in Computer Science* Vol. **1051**, 214--227, Springer Verlag, Berlin.

Jones, Cliff. 1996. Remarks during an invited talk
at the Northern Formal Methods Workshop, Ilkley, England.

McDermid, John. 1996. Remarks during an invited talk
at the Northern Formal Methods Workshop, Ilkley, England.

Paleska J. 1996. Test Automation for Safety Critical Systems: Industrial Application and Future Developments. In In Gaudel M.C. and Woodcock J. (eds). 1996. FME'96: Industrial Benefit and Advances in Formal Methods, *Lecture Notes in Computer Science* Vol. **1051**, 39--59, Springer Verlag, Berlin.

INDEX