

Proceedings of the
European **S**oftware **P**rocess **I**mprovement
Conference 2001

Eur SPI'2001

10.-12.10.2001, Limerick Institute of Technology, Limerick, Ireland

**Conference Theme 2001: The Combination of
Methodologies, Tools, And Cultural Factors**

Key Notes:

Mark Paulk, SEI, Pittsburgh, USA
C. Nökleby, GRUNDFOS, Denmark



Sponsored By



Supported By

WWW.LIT.IE

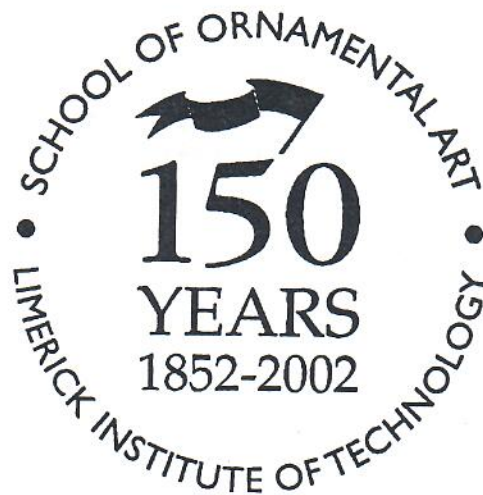
WWW.TECHNET.IE

**Proceedings of the European Software Process
Improvement Conference 2001**

EuroSPI 2001

**10th – 12th October 2001, Limerick Institute of Technology,
Limerick, Ireland**

**Conference Theme 2001: The Combination of Methodologies,
Tools and Cultural Factors.**



ISBN: 0-9541582-0-2

Published by Limerick Institute of Technology Press, Moylish Park, Limerick Ireland.
October 2001

Price: €25.00

Limerick Institute of Technology Press, Limerick Institute of Technology, Moylish Park, Limerick.
Tel: + 353 61 208 803 Fax: +353 61 208 209 E-mail: litpress@lit.ie

Programme Committee

Micheal Mac an Airchinnigh, Trinity College & ISCN, IRL
Gualtiero Bazzana, Onion Technologies, It
Miklos Biro, Sztaki, HU
Mads Christiansen, Delta, DK
Howard Duncan, Dublin City University, IRL
Taz Daughtrey, Chief Editor of the Software Quality Professional Magazine, USA
John Elliott, DERA, UK
Tim Hind, AXA-Sunlife UK
Bernd Hindel, ASQF, D
Theresa Hunt, Programs Chair of the ASQ Software Division, USA
Ioana Ene, Onion Technologies, It
Jorn Johansen, Delta, DK
Carsten Jorgensen, Delta, DK
Karl Heinz Kautz, Copenhagen Business School, DK
Patricia McQuaid, Americas Program Committee Chair for the Second (2000) and Third (2005) World Congresses on Software Quality, USA
Nils Brede Moe, Sintef, No
Messnarz Richard, ISCN IRL & ISCN A
Risto Nevalainen, STTF, Fin
Carsten Nøkleby, Grundfos A/S
Poul Grav Petersen, Novo A/S
Dag Sjøberg, University of Oslo, No
Terttu Orci, Royal Inst. of Technology & Stockholm University, S
Tor Stalhane, Norwegian University of Science and Technology, No
David Teague, British Telecom, UK
Timo Varkoi, Pori School of Technology, Fin
Yingxu Wang, University of Calgary, Canada

International Organising Committee

Richard Messnarz, ISCN, Ireland & Austria
Nils Brede Moe, Sintef, Norway
Risto Nevalainen, STTF, Finland
Jorn Johansen, Delta, Denmark
John Elliott, QinetIQ, UK
Bernd Hindel, Methodpark, D
Terttu Orci, University of Stockholm, Sweden

Local Organising Committee

John Donovan, Limerick Institute of Technology, Ireland
Adrienne Clarke, ISCN, Ireland

Marion Binot, ISCN, Ireland

Background

LIT as a local organiser sponsors by providing local facilities such as conference halls and tutorial rooms. "Limerick Institute of Technology is, at 150 years, the oldest of the Institutes of Technology in Ireland. Currently it has nearly 6000 students enrolled on a variety of programmes covering ICT, Biotechnology, Management Science, Built Environment among others. Over the last two years LIT has launched a major drive to promote research both within the Institute and in collaboration with local and national industry and state bodies. Research activity has doubled in each of the last two years and is expected to continue to grow over the next few years. Historically, the Institutes of Technology have underpinned regional industrial and cultural development. LIT has played an important role in establishing the Shannon Region as one of the most innovative and fastest growing regions in Ireland outside of Dublin. The EuroSPI conference represents a vote of confidence in the Institutes of Technology and we are delighted to be part of this international event."

Tecnet (The Technology Network Ireland) supports the promotion of the conference through its research networks and good connections to Enterprise Ireland and the Institutes of Technologies in Ireland. Tecnet provides Irish industry with research, development, and consulting services and technology transfers utilising the skills and facilities within all Institutes of Technology in Ireland.

Shannon Development is sponsoring the social event of the EuroSPI 2001 conference in the castle of Bunratty. They are a focal point if any organisation in this area plans joint ventures, business cooperations, and so forth. Shannon Development is today, Ireland's only dedicated regional development company. The Company's brief is to generate industry, tourism and rural development in the wider Shannon area, known as the Shannon Region. This covers an area of some 10,000 square kilometres spanning counties Clare, Limerick, North Tipperary, South Offaly and North Kerry, which collectively have a population of over 407,000 people. The Company's main areas of activity include;

- Developing and strengthening the indigenous industry sector in the Shannon Region,
- Developing the Shannon Free Zone as a location for international, Tourism marketing and new product development in the Shannon Region,
- Developing the Shannon Estuary as an integrated coastal zone and an international tourism and industrial location,
- Stimulating development at local and regional levels.

EuroSPI conferences present and discuss practical results from improvement projects in industry, focussing on the benefits gained and the criteria for success. Leading European industry are contributing to and participating in this event. This year's event is the 8th of a series of conferences to which countries across Europe and from the rest of the world contributed their lessons learned and shared their knowledge to reach the next higher level of software management professionalism.

INSERC is one of the newest research centres to be established in LIT (Limerick Institute of Technology), Ireland. INSERC2001- workshop in software ergonomics - takes place as a co-located event with EuroSPI

Session 1 - SPI and Human Factors

Session Chair:
Eugene O'Leary, Tecnet, Ireland

Experiences with the Impact of Cultural Factors on SPI	1-2
The Human Element in Process Improvement at Intel Technology Poland	1-17
PbP: A Programming Paradigm for Inexperienced Software Teams	1-23

Experiences with the Impact of Cultural Factors on SPI

Dr. Miklós Biró
MTA SZTAKI
Budapest, Hungary
miklos.biro@sztaki.hu

Dr. Richard Messnarz
ISCN
Bray, Co. Wicklow, Ireland
rmess@iscn.ie

Alfred Gerald Davison
GN Nettet A/S
Brøndby, Denmark
Gerald.Davison@gnnettest.com

Introduction

A value system is the deepest of four layers of culture considered in social anthropology. Layers and dimensions of national, organizational, and other group cultures have been identified by thorough scientific research. Layers of national culture from the most superficial to the deepest are briefly summarized here [Hofstede, 1994]:

Symbols: words, gestures, pictures, objects that carry a particular meaning which is only recognized by those who share the culture.

Heroes: persons, alive or dead, real or imaginary, who possess characteristics which are highly prized in a culture, and who thus serve as models for behavior.

Rituals: collective activities technically superfluous in reaching desired ends, however socially essential.

Values: broad tendencies to prefer certain states of affairs to others.

The first 3 layers are visible to an outside observer; their cultural meaning, however is invisible. The 4th layer, that is values are acquired so early in our lives, that they remain for the most part unconscious. Therefore they are not normally discussed, nor can outsiders normally directly observe them.

In order to illustrate the impact of value systems on views about the right management of organizations which are the subjects of process improvement, we refer to [Hofstede, 1994] lining-up four distinguished scholars from France, Germany, the US, and China. We first observe the differences in their value systems reflected by their views. We then briefly present Hofstede's model of national cultures, which we validate on its power in predicting the previously observed differences. We examine the impact of national cultural value systems on the effectiveness of process improvement models in general and on CMMISM¹(Capability Maturity Model (CMM[®]) IntegrationSM) in particular, whose continuous representation is compatible with the ISO/IEC 15504 (SPICE~Software Process Improvement and Capability dEtermination) model. We propose a 3rd cultural dimension in addition to the process and capability dimensions of the existing models. Finally, we observe the generic practices of CMMI from the cultural dimension.

It is obvious that the effectiveness of process improvement methods is also influenced by organizational culture. Nevertheless, the focus of this paper is on national culture.

Views about the right management of organizations

Henri Fayol (1841-1925) was a French engineer. His key work was *Administration Industrielle et Generale*, 1916. He belongs to the Classical School of management theory and was writing and exploring administration and work. The following quotation is from [Fayol, 1916] translated by G.Hofstede [Hofstede, 1994]:

“We distinguish in a manager his statutory authority which is in the office, and his personal authority which consists of his intelligence, his knowledge, his experience, his moral values, his leadership, his service record, etc. For a good manager, personal authority is the indispensable complement to statutory authority.”

It is clear that in Fayol's value system a person is a good manager if his power is both accepted by people and formally assigned by its organization.

Max Weber (1864-1920) was a German sociologist. He was the first to observe and write on the bureaucracy which developed in Germany during the 19th century. He considered it to be efficient, rational and honest, a big improvement over the haphazard administration it replaced. A quotation from [Weber, 1921]:

“The authority to give the commands required for the discharge of duties should be exercised in a stable way. It is strictly delimited by rules concerning the coercive means... which may be placed at the disposal of officials.”

¹ Capability Maturity Model and CMM are registered trademarks in the U.S. Patent and Trademark Office. CMM Integration and CMMI are service marks of Carnegie Mellon University.

In Weber's value system the management of an organization is good if it is strictly governed by rules. This is the original meaning of bureaucracy without the negative sense attached to it nowadays.

Mary Parker Follett (1868-1933) was a US pioneer of organization theory. In the 1920's, her comments and writing on leadership, power, law of the situation, conflict integration and circular behavior, empowerment, teams, and networked organizations, importance of relationships within and among organizations, authority, control, etc. were way ahead of her time. She writes [Metcalf, Urwick, 1940]:

“How can we avoid the two extremes: too great bossism in giving orders, and practically no orders given? ... My solution is to depersonalize the giving of orders to unite all concerned in a study of the situation, to discover the law of the situation and to obey that... One person should not give orders to another person but both should agree to take their orders from the situation.”

According to Follett an organization is well managed if it is governed by neither accepted nor formal power but the market situation in today's terms.

Dr. Sun Yat-sen (1866-1925) was a revolutionary ahead of his time. He devoted his life to bringing democracy to China. He extensively studied the political systems of European countries and America in formulating the Three Principles of the People: Nationalism, Democracy, and Social Well-being. After failing on several attempts to unite the people to revolt, Sun finally succeeded with the Wuch'ang Uprising on October 10, 1911, leading to the successful overthrow of the Ch'ing government and the establishment of the Republic of China.

Dr. Sun was a contemporary of the other scholars even if he did not address industrial but political organizations. The government structure of Taiwan builds on his ideas integrating the western separation of executive, legislative, and juridical powers with the Chinese tradition by making all of these dependent on the President, and adding an examination and a control power supposed to audit the government. Article Contents

Dimensions of national cultures

The seminal work [Hofstede, 1994] identifies the generic factors, which characterize value systems in different national cultures, including those of software and systems developers', applying statistical cluster analysis. The analysis was based on questionnaires from more than 50 countries. Each of the countries could be given an index score for each of the following dimensions of national cultures briefly described below:

Power distance characterizes the extent to which people consider it natural that power, status, and privileges are distributed unequally among individuals or that this distribution has no high significance in their lives. In small power distance countries subordinates and superiors consider each other, as existentially equal and decentralization is popular, while large power distance countries subscribe to authority of bosses and centralization.

Individualism versus collectivism characterizes people's esteem of individual activities and successes versus the importance of their belonging to a social group. In an individualist culture people are supposed to take care only of themselves and their immediate families, and remain emotionally independent from the group. In a collectivist

culture people distinguish between in-groups and out-groups, expect their in-group to look after them, and individuals define their identity by relationships to others and group belonging. The individual and the group have a mutual obligation of protection in exchange for loyalty.

Masculinity versus femininity is better expressed as confrontation and quantity orientation versus compromise and quality orientation. In masculine cultures importance is placed on assertiveness, competitiveness and materialism in the form of earnings and advancement, promotions and big bonuses. A feminine culture indicates the concern for people, the quality of life, nurturing and social well being.

Uncertainty avoidance characterizes people's attitude towards ambiguous or unknown situations. Innovation usually involves a lot of uncertainty; it is by consequence easier in weak uncertainty avoiding cultures. A strong uncertainty avoiding culture creates high anxiety in people who usually like to work hard and like establishing and following rules. The actual implementation of the results of innovation is an activity, which exactly requires this attitude.

All the four dimensions are a continuum between two extremes and no national culture is at one or the other extreme. Furthermore, the index scores are statistical results, which means there are always individuals which are not conform to the general model.

There is in fact a fifth dimension identified only later due to the natural western cultural bias of the experts themselves compiling the questionnaires used for the study. This is

Long-term versus short-term orientation or **Confucian dynamism** [Hofstede, 1994] which means persistence, establishment and observation of priorities, thrift, and a sense of shame on the long-term orientation pole, personal steadiness, protection of "face", respect for traditions, reciprocation of greetings, favors, and gifts on the short-term orientation pole.

The individualism-collectivism, and the masculinity-femininity dimensions are more oriented toward persons, while the power distance and uncertainty avoidance dimensions relate more to organizations.

Let us position the four scholars in these two organization-related dimensions.

Personal power as well as formal rules are important for Fayol whose culture is likely to have a high power distance and high uncertainty avoidance index.

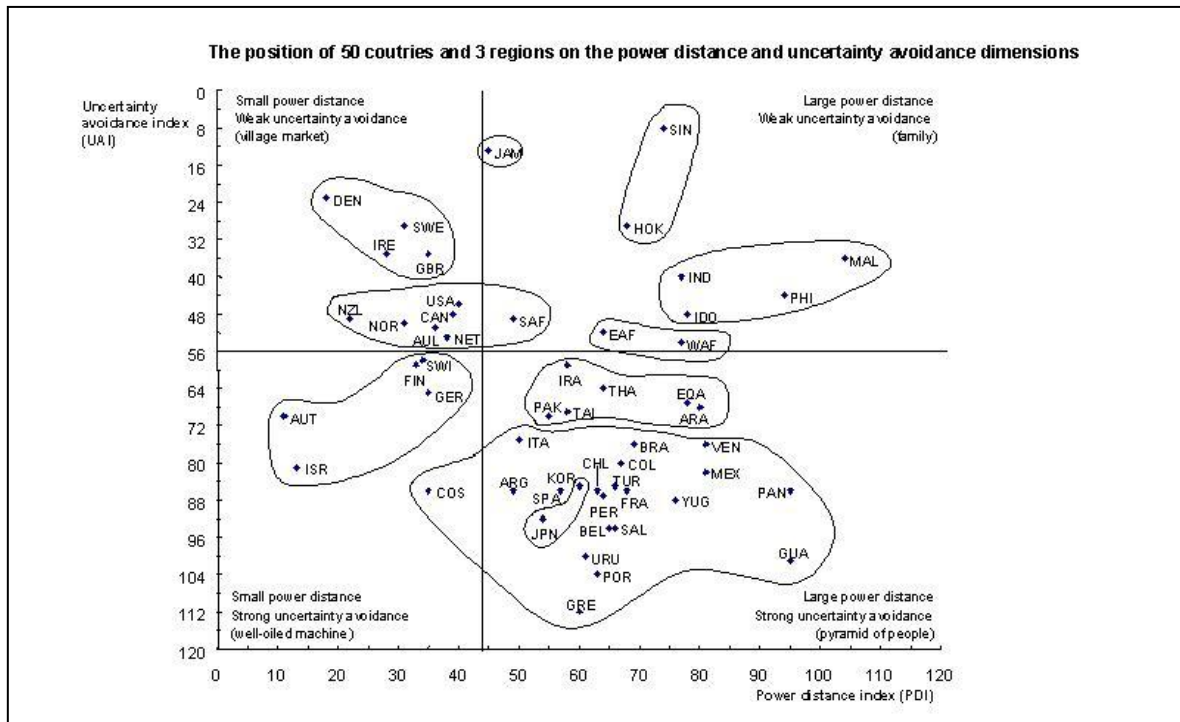
Power is not but rules are important for Weber whose culture is supposedly of low power distance and strongly uncertainty avoiding.

Follett rejects both power and rules, which lets us presuppose a low power distance, weak uncertainty avoiding culture.

Dr. Sun's political structure is built on power, which on the other hand is supposed to be controllable. These are characteristics of a large power distance, weak uncertainty avoiding culture.

Let us see Figure 1 with the position of 50 countries and 3 regions in the two dimensional space of power distance and uncertainty avoidance established in [Hofstede, 1994].

Figure 1



Abbreviation	Country	Abbreviation	Country
ARA	Arab countries (Egypt, Iraq, Kuwait, Lebanon, Libya, Saudi Arabia, United Arab Emirates)	JPN	Japan
ARG	Argentina	KOR	South Korea
AUL	Australia	MAL	Malaysia
AUT	Austria	MEX	Mexico
BEL	Belgium	NET	Netherlands
BRA	Brazil	NOR	Norway
CAN	Canada	NZL	New Zealand
CHL	Chile	PAK	Pakistan
COL	Colombia	PAN	Panama
COS	Costa Rica	PER	Peru
DEN	Denmark	PHI	Philippines
EAF	East Africa (Ethiopia, Kenya, Tanzania, Zambia)	POR	Portugal
EQA	Ecuador	SAF	South Africa
FIN	Finland	SAL	Salvador
FRA	France	SIN	Singapore
GBR	Great Britain	SPA	Spain
GER	Germany FR	SWE	Sweden
GRE	Greece	SWI	Switzerland
GUA	Guatemala	TAI	Taiwan
HOK	Hong Kong	THA	Thailand
IDO	Indonesia	TUR	Turkey
IND	India	URU	Uruguay
IRA	Iran	USA	United States
IRE	Ireland (Republic of)	VEN	Venezuela
ISR	Israel	WAF	West Africa (Ghana, Nigeria, Sierra Leone)
ITA	Italy	YUG	Yugoslavia
JAM	Jamaica		

By examining the position of France, Germany, the US, and countries with strong Chinese cultural influence like Hong Kong and Singapore in the four quadrants of Figure 1, we can clearly conclude as Hofstede did, that his model is powerful in predicting differences in views about the right management of organizations among others.

The 3rd Dimension

The following discussion is based on the CMMI framework, it can however obviously be applied to any method intending to improve the processes of an organization.

CMMI is a (see Figure 2) framework from which models can be generated for different organizations. “Although process areas depict behavior that should be exhibited in any organization, practices must be interpreted using an in-depth knowledge of the CMMI model, the organization, the business environment, and the specific circumstances involved.”[CMMISM-SE/SW/IPPD, V1.02 Continuous Representation/lines 559-562]

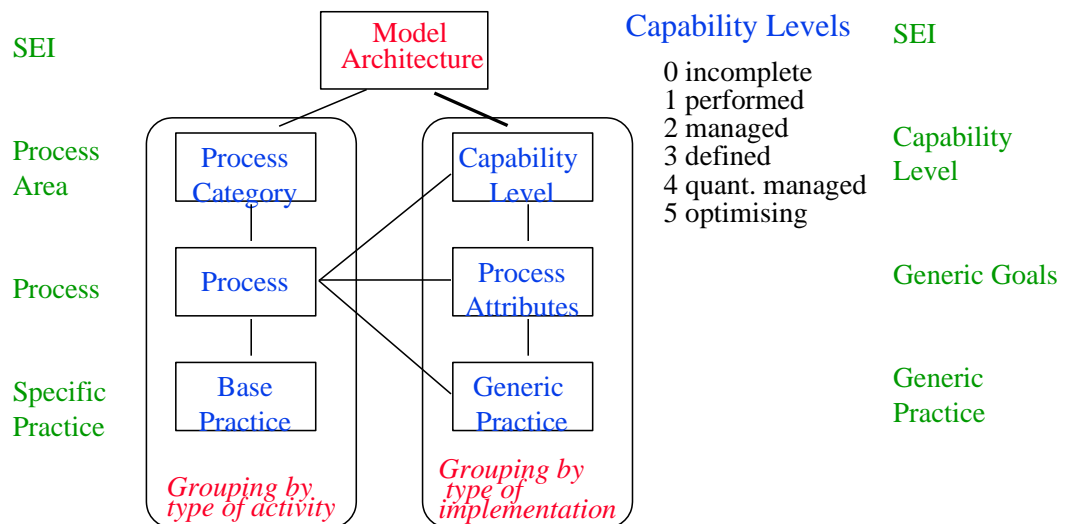


Figure 2

The above statement bears a striking resemblance to Mary Parker Follett’s way of thinking which is not surprising because of the obvious fact that CMMI was developed in the U.S. cultural environment. It is fortunate that at the same time, this statement clearly demonstrates the intention to be open to considerations regarding all kinds of circumstances including differences in cultural value systems.

CMMI Continuous Representation is often illustrated as having a two dimensional structure with the process dimension on one side and the capability dimension on the other. This is a sound approach, since each (process area, capability level) pair has a well defined meaning and implies suitable improvement actions.

With this paper, we propose to work towards a 3rd dimension (see Figure 3) in the CMMI architecture: **the cultural dimension**. This model extension is valid, since the national cultural position of the company may determine a different meaning and suitable improvement actions for every (process area, capability level) pair, or even every (process area, specific or generic practice) pair using a finer granularity of CMMI where

process capability levels are achieved by performing the specific or generic practices on the process areas. In short, the effectiveness of various practices depends on the national culture where they are performed. In fact, this model extension can be fully accommodated by even the current version of the CMMI since all practices are only expected and not required model components [CMMISM-SE/SW/IPPD, V1.02 Continuous Representation/lines 824-825].

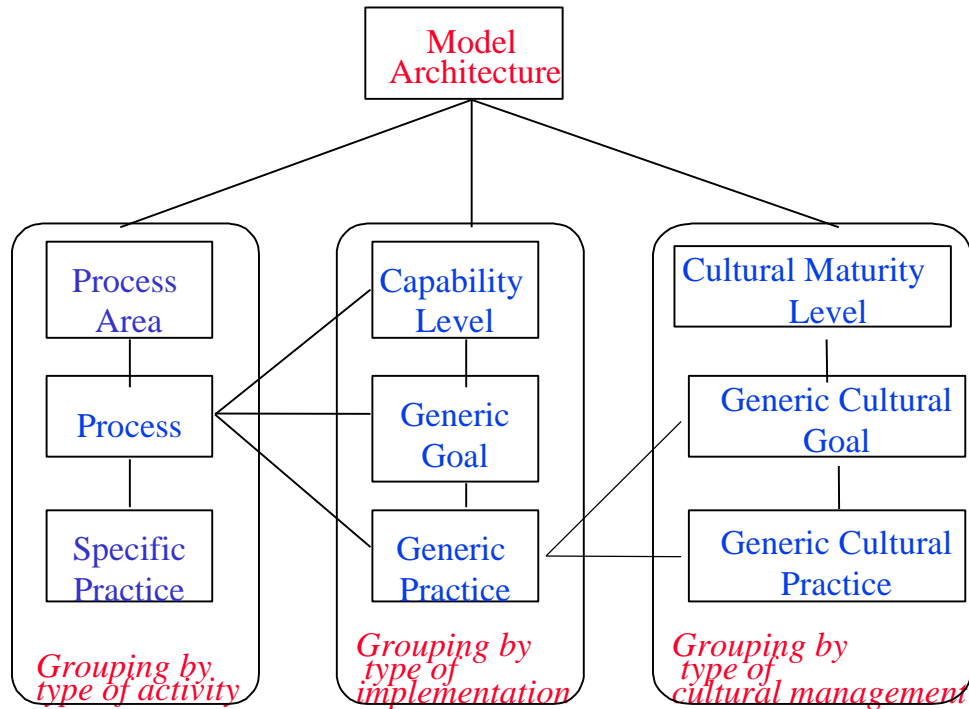


Figure 3

We have seen that the cultural dimension itself is multidimensionally determined by Hofstede but there are other models of national cultures which are also valid and which we intend to leave the new model open to.

In order to remaining consistent as far as possible with the structure of the CMMI model, the new cultural dimension has model components similar to those of the capability dimension of CMMI. Nevertheless, we use the old maturity level terminology for the cultural dimension, since it has a better descriptive power in this case. The model has also similarities with the ancestor of CMM [Paulk, 1995] [Messnarz, Tully, 1999]: Crosby’s Maturity Grid [Crosby, 1979].

Cultural Maturity Level 0 (Closed)

The cultural maturity level of a process area at a given capability level is 0 if the specific and/or generic practices, leading to the achievement of the specific and/or generic goals of the process area at the given capability level, are prescriptive to the extent where no differences in cultural value systems are allowed.

Cultural Maturity Level 1 (Open)

The cultural maturity level of a process area at a given capability level is 1 if the specific and/or generic practices, leading to the achievement of the specific and/or generic goals of the process area at the given capability level, are open enough to allow for differences in cultural value systems.

Cultural Maturity Level 2 (Model based)

The cultural maturity level of a process area at a given capability level is 2 if the consideration of cultural differences is based on a scientifically established model. Hofstede's multidimensional model is an example, but other models are also acceptable in case they are useful in distinguishing the differences in cultural value systems which have an impact on the performance of the specific and/or generic practices.

Cultural Maturity Level 3 (Comprehensive)

The cultural maturity level of a process area at a given capability level is 3 if the scientifically established cultural model is comprehensively applied to all specific and generic practices leading to the achievement of the specific and/or generic goals of the process area at the given capability level.

Cultural Maturity Level 4 (Tailored)

The cultural maturity level of a process area at a given capability level is 4 if the depth and complexity of the application of the cultural model to the specific and generic practices is based on quantitatively managed experience and business needs.

Cultural Maturity Level 5 (Competency driven)

The cultural maturity level of a process area at a given capability level is 5 if the cultural model applied to the specific and generic practices is refined, extended, or fully changed on the basis of competency acquired through quantitatively managed long-term model experience and business needs.

Level 1 Generic Cultural Goal

Create an organizational culture where specific and generic practices allow for differences in cultural value systems when the need is identified.

A level 1 Generic Cultural Practice is to scan specific and generic practices for cultural restrictions and relieve the identified restrictions.

An excellent example of a practice in CMMI where cultural restrictions are already partially relieved is Generic Practice 2.4 (Assign Responsibility) [CMMISM-SE/SW/IPPD, V1.02]. The description of the practice contains the following wording: "Responsibility can be assigned using detailed job descriptions..." Strongly uncertainty avoiding German engineers will love this approach as opposed to weakly uncertainty avoiding Swedish engineers. GP 2.4 states however that "Dynamic

assignment of responsibility is another legitimate way to perform this practice...". We can claim by consequent that GP 2.4 can actually be adapted to both weakly and strongly uncertainty avoiding national cultures.

Level 2 Generic Cultural Goal

Specific and generic practices take cultural differences into consideration on the basis of a scientifically established model.

A level 2 Generic Cultural Practices is the application of an element of the cultural model to selected specific and generic practices.

In the case of the Hofstede model, Generic Cultural Practices (GCP) could be the following:

GCP 2.1 Consider the power distance factor in selected specific and generic practices.

GCP 2.2 Consider the the individualism versus collectivism factor in selected specific and generic practices.

GCP 2.3 Consider the masculinity versus femininity factor in selected specific and generic practices.

GCP 2.4 Consider the uncertainty avoidance factor in selected specific and generic practices.

GCP 2.5 Consider the long term versus short term orientation factor in selected specific and generic practices.

In CMMI Generic Practice 2.4 of the above example, the consideration of the individualism versus collectivism factor of the Hofstede model (GCP 2.2) leads to another cultural restriction of this practice which should be relieved.

The assignment of responsibility is strongly related to the individualism versus collectivism dimension. Should the responsibility be assigned to people or rather teams? The wording of GP 2.4 reflects cultural conditioning:

“Confirm that the people assigned to the responsibilities and authorities understand and accept them.” [CMMISM-SE/SW/IPPD, V1.02 Continuous Representation /lines 1744-1745]

In order to illustrating the importance of this issue, we quote [Hofstede, 1994] referring to a management researcher from the U.S., Christopher Earley who performed an enlightening laboratory experiment on a group of 48 management trainees from southern China and 48 matched management trainees from the U.S. Half of the participants in either country were given group tasks, the other half individual tasks. Also, half of the participants in either country, both from the group task and from the individual task subsets were asked to mark each completed item with their names, the other half turned them in anonymously. “The Chinese collectivist participants performed best when operating with a group goal and anonymously. They performed worst when operating individually and with their name marked on the items produced. The American individualist participants performed best when operating individually and with their name marked, and abysmally low when operating as a group and anonymously.”

In addition to GP 2.4 and many other parts of CMMI, the above experiment has an obviously profound impact on such IPPD process areas like Organizational Environment for Integration and Integrated Teaming.

Level 3 Generic Cultural Goal

The scientifically established cultural model is comprehensively applied to all specific and generic practices.

A level 3 Generic Cultural Practice is the systematic application of an element of the cultural model to all specific and generic practices.

In the case of the Hofstede model, Generic Cultural Practices (GCP) could be the following:

GCP 3.1 Consider the power distance factor in all specific and generic practices.

GCP 3.2 Consider the the individualism versus collectivism factor in all specific and generic practices.

GCP 3.3 Consider the masculinity versus femininity factor in all specific and generic practices.

GCP 3.4 Consider the uncertainty avoidance factor in all specific and generic practices.

GCP 3.5 Consider the long term versus short term orientation factor in all specific and generic practices.

CMMI Generic Practice 2.8 (Monitor and Control the Process) and Generic Practice 2.10 (Review Status with Higher-Level Management) are both affected by GCP 3.1 and GCP 3.4. CMMI Generic Practice 2.7 (Identify and Involve Relevant Stakeholders) is on the other hand affected by GCP 3.3, and Generic Practice 3.2 (Collect Improvement Information) by GCP 3.5.

GCP 3.1 (Power distance) considered in GP 2.8 (Monitor and Control the Process) and GP 2.10 (Review Status with Higher-Level Management)

Both of these generic practices require a review involving communication with either the immediate level of management or higher-level management. Power distance has a determining impact on the communication considered appropriate in a given culture.

Referring again to [Hofstede, 1994] let us quote a senior Indian executive with a Ph.D. from the U.S. [Negandhi, Prasad, 1940]:

“What is most important for me and my department is not what I do or achieve for the company, but whether the Master’s favor is bestowed on me. ... This I have achieved by saying “yes” to everything the Master says or does. ... To contradict him is to look for another job. ... I left my freedom of thought in Boston.”

It is obvious now that the way of performing generic practices 2.8 and 2.10 must take into account the power distance index in the national culture where the organization is located.

GCP 3.4 (Uncertainty avoidance) considered in GP 2.8 (Monitor and Control the Process) and GP 2.10 (Review Status with Higher-Level Management)

"Track corrective action to closure" is an important subpractice of GP 2.8. Precision and punctuality required by this subpractice is a natural characteristic of strongly uncertainty avoiding cultures, which will by consequent be better in this respect.

GCP 3.3 (masculinity versus femininity) considered in GP 2.7 (Identify and Involve Relevant Stakeholders)

The paper [Atwong, Lange, 1996] gives account of a virtual classroom experiment with students of the California State University-Fullerton and Lappeenranta University of Technology, Finland. The subject of the experiment was a marketing research project, which is irrelevant in our context. The important is that “the project combined the American and Finnish students into one virtual classroom with cross-national teams. Students used the Internet extensively for data collection... and conducted Internet chat with foreign team members when necessary.” The message of the story can be summarized with the opinion of a Finnish student: "It was interesting to see the effect of cultural differences, even in a relatively simple project like this. When we first established contact with our American teammates, they wanted first to introduce themselves and chat about their interests and hobbies, which we thought was strange. Later we realized that this was their way to establish rapport with small talk. The Finns are used to getting immediately down to business. In the oral presentations, the American students seemed to emphasize presentation technologies more than us. However, in my opinion the quality of the work was roughly equal."

The above observation is due to the difference between the U.S. and Finland on the masculinity versus femininity scale, which is the only dimension where the U.S. and Finland are significantly different. The involvement and the resolution of the conflicts of stakeholders from even these two otherwise close cultures requires a careful handling of the cultural differences without which both the assertive U.S. students and the modest Finnish students may find each other ridiculous, strange, shocking or even hateful.

This issue has of course to be taken into account while stakeholder involvement is planned in the Project Planning process area.

GCP 3.5 (long term versus short term orientation) considered in GP 3.2 (Collect Improvement Information)

Process improvement as a whole and especially this GP 3.2 clearly requires long-term orientation, that is persistence, establishment and observation of priorities, and thrift. Short-term orientation, that is protection of “face”, respect for traditions act against process improvement.

Level 4 Generic Cultural Goal

Experiences with the consideration of cultural differences are quantitatively managed, and the depth and complexity of the application of the cultural model is based on the quantitatively managed experiences and business needs.

Experience may indicate the need for a deeper application of the Hofstede model. In fact, Hofstede himself examined pairs of cultural factors in addition to the single ones as demonstrated on Figure 1 of this paper. GCP 4.1 considers such a pair for example. The number of pairs of the 5 cultural dimensions is $5C2 = 10$. We have to be careful however with increasing the number of considered combinations, since the number of cases may increase exponentially which is not useful for our purposes.

GCP 4.1 Consider the pair of power distance and uncertainty avoidance cultural factors in all specific and generic practices.

CMMI Generic Practice 2.7 (Identify and Involve Relevant Stakeholders) will be affected by GCP 4.1.

GCP 4.1 (power distance and uncertainty avoidance) considered in GP 2.7 (Identify and Involve Relevant Stakeholders)

The preferred way of resolving conflicts between stakeholders can be predicted from the position of a culture in the two dimensional space of power distance and uncertainty avoidance shown on Figure 1 just as the views of the distinguished scholars.

[Hofstede, 1994] describes the results of an organizational behavior course examination reported by Owen James Stevens, an American professor at INSEAD business school in Fontainebleau, France. A mixture of French, German, and British students received a case study where they had to resolve a conflict between two department heads within a company. A sales and a manufacturing manager for example have usually conflicts since sales tries to satisfy changing customer demands, while manufacturing is more efficient if batches are larger and changes are less frequent.

“The results were striking. ... The solution preferred by the French was for the opponents to take the conflict to their common boss, who would issue orders for settling such dilemmas in the future. ... The solution preferred by the Germans was the establishment of procedures.” The British solution was the registration of both department heads to a management course to develop their negotiation skills.

In summary, the French with large power distance and strong uncertainty avoidance prefer to concentrate the authority and structure the activities, the Germans with strong uncertainty avoidance but smaller power distance want to structure the activities without concentrating the authority, while the British with small power distance and weak uncertainty avoidance believe in resolving conflicts ad hoc.

Level 5 Generic Cultural Goal

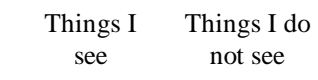
The cultural model is refined, extended, or fully changed on the basis of competency acquired through quantitatively managed long-term model experience and business needs.

For example, there may be a need for the consideration of cultural factors which are impractical to evoke from the existing cultural model. Below are the discussions of Generic Cultural Practices for two such factors.

GCP 5.1 Consider the discrepancy in perceived understanding

In [O’Suilleabhain 2000], a practical example of the Johari window is described showing how discrepancy in perceived understanding is measured in the “OSIRIS” project.

The Johari window is a tool originally designed for conceptually distinguishing between 4 different possible states with regard to knowledge of oneself, these states are shown in the diagram below:



Open for discussion	My blind spot	Things they see
Their blind spot	Shared blind spot	Things they do not see

Figure 4

The same principle can be applied to measure 4 different possible states with regard to knowledge of one’s culture, and the understanding of other cultures in the project team.

Respondents to the OSIRIS survey were asked to rate, on a scale from 1 to 5, the discrepancy between the understanding cultures in the ISIS project have of themselves, and the understanding others have of them. The questions asked were based on the Johari Window.

The table below shows the perceived discrepancies between each culture’s understanding of itself and the understanding other cultures have of it. Note that the ratings for each culture are by the other two cultures not by own culture.

	<i>N</i>	<i>Mean</i>	<i>Std. Deviation</i>
Discrepancy between German respondent's own understanding of themselves and that of respondent	4	3.7500	.5000
Discrepancy between Irish respondents' own understanding of themselves and that of respondent	12	2.3333	.7785
Discrepancy between Greece's respondents' own understanding of themselves and that of respondent	10	2.4000	.8433

Figure 5

The bigger the discrepancy the smaller the incidence of perceived shared understanding and, implicitly then, the fewer issues “open for discussion”. It is interesting that Germany was the only culture with an above-average mean score which may imply that other cultures more often than not find their own understanding of German culture to be at odds with that of the Germans’ own and that “blind spots”, which can hinder effective interaction, may be greater than the Germans themselves believe.

A typical sign when this Johari problem happens is that during a certain project some results have to be re-agreed a number of times although the Germanic group feels that clear definitions have been agreed already.

This problem can be solved by moderating workshops in a way which allows to explain the different viewpoints and identify synergies, finally resulting in common solutions.

GCP 5.1.German. Do the managers understand that the once agreed business model might change driven by stakeholder inputs and new demands, and that cultural difference requires understanding different viewpoints on the same.

GCP 5.2 Consider the SPI approach in relationship with company culture and size.

As outlined in [Davison 2001] the movement of a large multinational organisation towards higher capability is a long lasting agreement process where step by step you must win the confidence of the managers and staff. Many tactics are needed to achieve this.

Example:

GCP 5.2.Large. Are the stakeholder agreements achieved step by step so that all involved parties are convinced and the joint mission is clear.

Conclusion

CMMI intends to be effective in all national cultures. By consequence, it should consider the way its practices can most effectively be performed in different national cultural environments. This paper draws attention to the deep relevance of this issue to the international success of CMMI and of all other process improvement methods.

References

- Atwong, C.T., Lange, I.L. (1996). How collaborative learning spans the globe, *Marketing News*, 8/12/1996, Vol.30 Issue 17, pp16-17.
- Biró, M. (2000). Cultural Environment Protection in the Information Society. In: *Project Control: The Human Factor, Proceedings of the combined 11th European Software Control and Metrics Conference and the 3rd SCOPE Conference on Software Product Quality* (ed. by K.D.Maxwell, R.J.Kusters, E.P.W.M.van Veenendaal, A.J.C.Cowderoy). (Shaker Publishing B.V., 2000) (ISBN 90-423-0102-3) pp.415-421.
- Davison, A.G. (2000). The Process Psyops – Moving a Large Company Towards SPI, In: *Proceedings of the EuroSPI2001 Conference* (eds. by B.Hindel, C.Jorgensen, J.Elliot, M.Christiansen, R.Messnarz, R.Nevalainen, T.Stalhane, Y.Wang). (Limerick Institute of Technology, Limerick, Ireland), 10-12.10.2001.
- Crosby, P.B. (1979). *Quality is Free – The Art of Making Quality Certain*. McGraw-Hill, New York, 1979.
- Fayol, H. (1916). *Administration industrielle et générale*, Dunod, Paris, 1916.
- Hofstede, G. (1994). *Cultures and Organizations, Software of the Mind: Intercultural Cooperation and its Importance for Survival*, McGraw-Hill, London, 1994.
- Metcalf, H.C., Urwick, L. (1940). *Dynamic Administrations: The Collected Papers of Mary Parker Follett*, Harper & Row, New York, 1940.
- Messnarz R. Tully C. (1999). *Better Software Practice for Business Benefit – Principles and Experience*. IEEE Computer Society Press, Brussels, Washington, Tokyo, 1999, ISBN : 0-7695-0049-8.

- Negandhi, A.R., Prasad, S.B. (1971). *Comparative Management*. Appleton-Century-Crofts, New York, 1971.
- O'Suilleabhain, G., Messnarz, R., Biró, M., Street, K. (2000). The Perception of Quality Based on Different Cultural Factors and Value Systems. In: *Proceedings of the EuroSPI'2000 Conference* (ed. by B.Hindel, C.Jorgensen, J.Elliot, M.Christiansen, R.Messnarz, R.Nevalainen, T.Stalhane, Y.Wang). (Copenhagen Business School, Copenhagen, Denmark, 2000) (ISBN 952-9607-29-6) pp.2-32 – 2-45.
- Paulk, M.C. (1995). The Evolution of the SEI's Capability Maturity Model for Software. *Software Process Improvement and Practice*, Vol.1. Pilot Issue, Spring 1995, pp.3-15.
- Siakas, K.V., Balstrup, B. (2000). A Field-study of Cultural Influences on Software Process Improvement in a Global Organisation. In: *Proceedings of the EuroSPI'2000 Conference* (ed. by B.Hindel, C.Jorgensen, J.Elliot, M.Christiansen, R.Messnarz, R.Nevalainen, T.Stalhane, Y.Wang). (Copenhagen Business School, Copenhagen, Denmark, 2000) (ISBN 952-9607-29-6) pp. 2-20 – 2-31.
- Weber, Max. (1921). *Essays in Sociology*. (ed. by H.H.Gerth and C.W.Mills), Routledge & Kegan Paul, London, 1948. (Translated from *Wirtschaft und Gesellschaft*, 1921)

The Human Element in Process Improvement at Intel Technology Poland

Piotr Umiński
*Intel Technology Poland,
Słowackiego 173, Gdańsk, Poland
Piotr.Uminski@intel.com*

Introduction

The success of a software process improvement initiative depends upon several factors, both technical and non-technical. The technical and organizational aspects of process improvements are well described in, for example, a form of popular Capability Maturity Model for Software [2], as well as by other models. However, it seems that the human element is a key to the success of software process improvement initiatives. Using the example of Intel Technology Poland, this paper illustrates the importance of the human element in raising process maturity.

Characteristic of the company

Intel Technology Poland (ITP), based in Gdańsk, Poland, is a middle-size R&D site owned by Intel Corporation. It employs more than 100 engineers, most of them software engineers, to produce software for network and telecommunication devices.

The R&D team in Gdansk was organized in 1991 as a subsidiary of CrossComm Corporation, U.S.A., to write software for CrossComm' flagship products: high-speed heterogeneous bridges and, later, modular routers. In 1997, Danish networking company Olicom A/S acquired CrossComm Corporation with its Polish subsidiary, where the Polish team immediately began a rapid development project for Ethernet L2 and L3 switches. In 1999, Intel Corporation purchased Olicom's R&D department intact.

Our team is young but quite experienced. The average engineer has been 5 years with the

organisation; a number of them have been with the team for 8 or 9 years. All of them are well educated (all have at least a M.Sc.) and have a much experience in networking and telecommunication research and development. The staff turnover rate at the site is very low.

Process improvements at ITP

All development projects in which this site was involved were more or less distributed between different sites. The hardware design and prototyping was done in the U.S or Denmark. Some software projects were developed by more than one development site as a joint effort, and some of them were developed mostly in Gdansk. Therefore, the organisation is experienced in cross-site development, and the importance of a standardised development process has always been clear.

The site's development process was first formally assessed in 1999 by Danish company Delta [4] using a BOOTSTRAP methodology. BOOTSTRAP [1] is a European method of process measurements that is based on the American CMM model [2][3]. Using BOOTSTRAP algorithm version 2.3, the assessed overall maturity level for the entire Danish and Polish Olicom R&D organisation was calculated to be 1.75, and was almost equal for Polish and Danish teams. This score, as defined in CMM, placed us a little below the industry average and far below our ambitions, and was a shock for us because the organization had already used many well-defined procedures and practices to develop several cross-site projects.

After the unsatisfactory result of the first assessment, we moved from ad hoc process improvements to an action program driven by a person dedicated only to process definition and improvement. Process improvement starts from education: site management learned about the CMM model and recommended practices, after which they decided that a reasonable long-term goal for our organization would be reaching CMM Level 3. Then started the long effort of recording our existing procedures and defining the missing ones.

The effort was sped up when Intel acquired the R&D team of Olicom and Intel's well-defined Product Life Cycle (PLC) was adopted. Intel's PLC defines standard documents, milestones, and meetings for each product developed by Intel. It was not difficult for ITP to adopt this standard and its terminology, since most of the similar documents and milestones were already used in previous projects. However, using the Intel PLC we got detailed written definitions of standard documents, milestones, and terms free.

The process standardisation effort was also driven by the requirements of large new cross-site projects developed at ITP under Intel. Effective collaboration between sites must be based on defined procedures for reaching consensus on requirements and deliverables as well as on standardised status reporting.

In March 2001, we had another process assessment (though this time only the Polish site was assessed) to get an independent assessment of our improvements. To ensure that the assessments were comparable, we again had Delta apply the same BOOTSTRAP algorithm (2.3) to our organisation. This time, the assessed maturity level was 3.25 (see Fig. PUM-1), which means that the organization has reached its goal within 1.5 year and that its maturity level is higher then the industry average. In the next section, we try to explain this success.

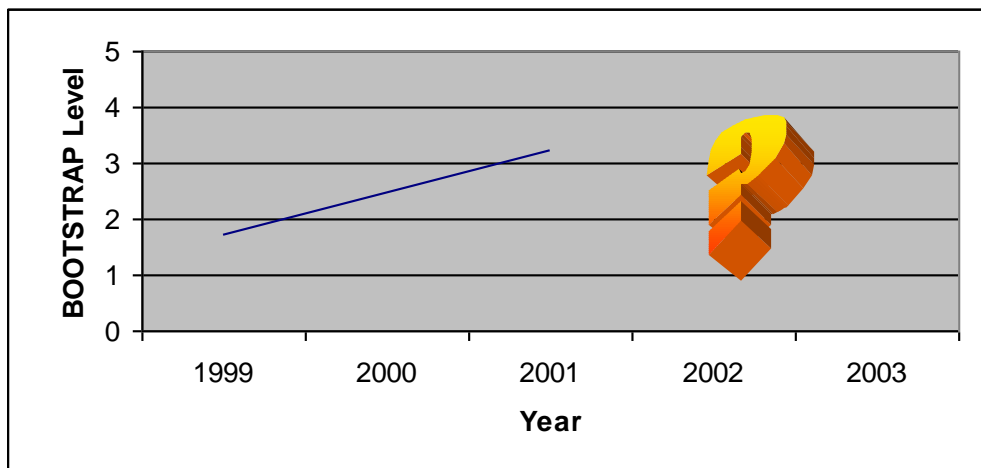


Fig. PUM-1. Process improvement at Intel Technology Poland (ITP).

The human element of the process improvement

Institutional memory

Institutional memory is a very important element of development process improvement. The staff turnover rate at ITP is very low and institutional memory very strong; many people have worked together through success and failure over a number of years, and have evolved their own working methods. We have noticed that experienced people better understand the role of a standardised process. This is perhaps because experienced employees can draw upon their experience of times when lack of proper standardisation created serious problems in their team efforts, while recent graduates sometimes come to us with bad habits learned from ad hoc student project development. Fortunately, in our environment the relatively few newcomers learn quickly to adopt the methods of the old-timers.

The challenge of acquisition

As previously stated, our R&D site has twice in the recent past changed hands as a larger company acquired the site's parent company. The acquisition and integration process is always a challenge for both organizations. The new owner may want to change everything in the acquired organization to unify them with the rest of the new organization. Such unification may include changing the management, procedures, standards, names, and colours of walls and furniture. Another possible situation is that the newly acquired organization may negate any changes towards the integration – so they use old templates, procedures, and names, and refuse to change anything. Both cases are examples of “Not-Invented Here” syndrome. And in both cases, the integration process is rendered less effective.

To avoid such inefficiency, during both transitions of the R&D site, two primary goals were to:

- Preserve and re-use the knowledge and experience embodied in the site

- Improve the technology and learn from the new organization

The ITP organization and the people within it are open-minded. They were always willing to learn and adopt a new practice proposed by the new owner— when they saw that it was a good practice. The importance of using certain common terminology is well understood. However, new practices were not accepted without criticism. When an existing procedure seemed to be better than a new one, a new one was rejected or at least improved before introduction. This was possible because, in most cases, procedures used within a site can be more formal and restrictive than used company-wide, as long as some standard interfaces (such as major milestones or essential documents) are common to them.

Process measurements are not for people judgment

Many essential process activities (such as defect tracking and reviews) can be seen as a method to control people rather than the product. In addition, advanced practices such as the usage of various metrics may be seen as a method to measure individual or team performance. The potential for conflict during process improvements is obvious. For example, when people fear being punished for too many open defects, they could stop using the defect tracking system and start communicating about defects by telephone and notepaper. If that happened, not only would the metrics be invalidated, but the defect-tracking process would fail and the number of unfixed defects in the released product (and therefore the number of unsatisfied customers) would very likely increase dramatically.

To avoid such a situation, from the very beginning we worked hard to show engineers that defect tracking, the change management process, and metrics would not be used to judge them individually or as teams. In Intel, there is a separate process to measure individual achievement and improvement. The process is defined by Human Resources, not R&D, and R&D statistics data about defects are not used in this process.

Education

Almost any changes performed in any situations create short-term difficulties. Therefore, the reason for changes has to be made clear for everyone involved. This is obvious in process improvement initiatives, in which formalising existing practices or adding new ones might be seen as adding burdens to developers while schedules are tight and resources are scarce. Engineers must therefore understand that a new initiative should improve the quality of the product and reduce the total effort, even if it creates short-term overhead.

We have seen that it is possible to rapidly design and introduce a new practice, but only when combined with longer-term education, because people need to know not only their own tasks, but also to understand the entire development process. For example, all engineers should know how product requirements are defined and why requirement management is an essential element of development, so they will understand the reasons for the changes they implement. When engineers understand the purpose of a milestone, they can deliver exactly what is expected before the milestone. In addition, of course, employee knowledge about the entire development process and common terminology helps to obtain better results during process audits and assessments. In our opinion, the poor result of our first assessment was caused by a lack of education: some engineers did

not know about some management practices or did not properly understand some terms.

The role of team morale

We mention team morale last, but it is, in fact, the essential element for successful process improvement. Members of our team identify themselves with the organization and products. They are very proud of the products we have created. They see process improvement as a tool that can help them to develop better products. Process improvement is therefore solidly supported by our engineers. For example, results of the last internal process assessment show that 76% of the developers are satisfied with a new and very rigorous coding standard, that 19% would prefer even more rigour, and that only 5% think that the new standard is too restrictive (see Fig. PUM-2). This is in contrast with the stereotype of the programmer who is against all restrictions to the programmer's coding freedom. Our engineers do indeed remain strong-willed and confident in their own engineering skills, but they understand the role of unification and teamwork in the successful development of large projects.

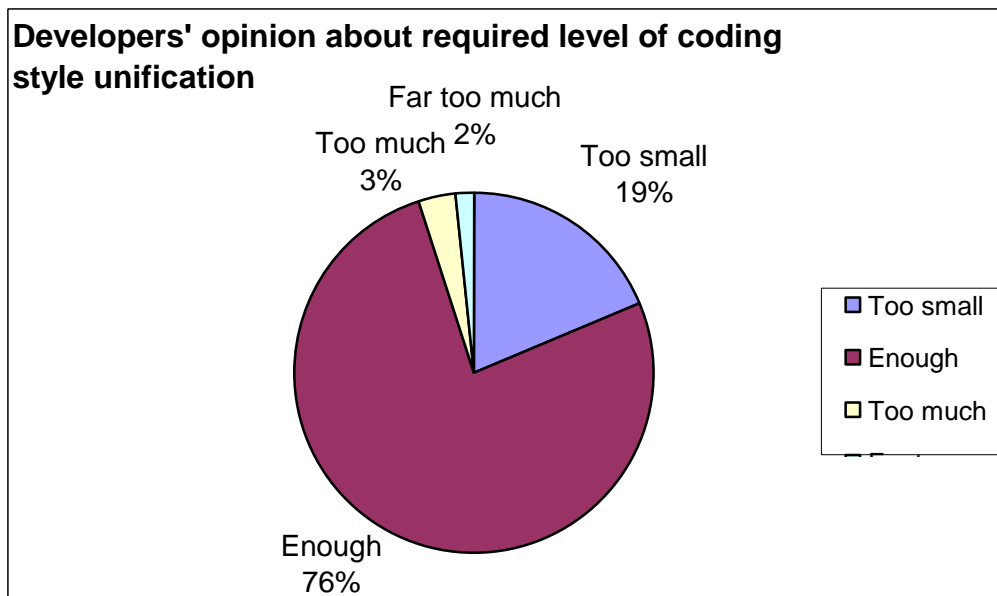


Fig. PUM-2. Result of internal process audit at ITP – developers' opinions about the requested standardization of the code

Our site has always enjoyed great team spirit and good manager-engineer relations. Many people at ITP have known each other since their university days, have worked hard together, and even spend leisure hours together. It is natural, therefore, that engineers and managers know and trust one another. Site management, however, works to support and develop this atmosphere through company parties, sporting competitions, and other group activities that ensure high spirits even in difficult times.

Summary

We found that rapid improvement of the software development process is possible. But we also found that it is not an easy task, and that, at least in the computer industry, it is

much easier to change technologies than to change minds.

The following key practices can help the organization to raise process maturity:

- Create and maintain institutional memory of success and failures in previous projects.
- Be open minded and analytical, challenge existing processes and learn from others experience, establish win-win situations, and avoid the negative effects of “Not Invented Here” syndrome.
- Put training in perspective; people learn and adopt new methods faster when they can see how things fit together.
- Keep a watertight wall between the personal performance metrics of Human Resources and the process and product metrics of R&D.
- Establish and maintain a social network between employees that focuses on human values different from professional values.

In our organization, we have recognized that the human element is both the most difficult and the most important element of process improvement, so we were able to improve the process quickly and efficiently.

References

- [1] P. Kuvaja, J. Simila, L. Krzanik, A. Bicego, S. Saukkonen, G. Koch: Software process assessment and improvement: the BOOTSTRAP approach, Blackwell Business, Oxford, UK, and Cambridge, MA 1994
- [2] M. C. Paulk, B. Curtis, M. B. Chrissis, C. V. Weber: Capability Maturity Model for Software, Version 1.1, Software Engineering Institute Technical Report CMU/SEI-93-TR-024
- [3] M. C. Paulk, C. V. Weber, S. W. Garcia, M. B. Chrissis, M. Bush: Key Practices of the Capability Maturity Model, Version 1.1, Software Engineering Institute Technical Report CMU/SEI-93-TR-025
- [4] DELTA Danish Electronics, Light and Acoustics web pages: www.delta.dk

PbP: A Programming Paradigm for Inexperienced Software Teams

Kim Man Lui

The Hong Kong Polytechnic University

Keith CC Chan

The Hong Kong Polytechnic University

Introduction

Plagiarism is an act that is considered unacceptable by many. In this paper, however, we describe a programming paradigm based on it and we encourage inexperienced programming teams to adopt it. This paradigm, which we call PbP (Plagiarism based Programming), can be interpreted as an approach to write software programs so as to ensure that these programs, or part of these programs, can be easily copied or modified for re-use. By re-use, it means, in this context, both reusability of patterns and repeatability of source code by people. The former is concerned with patterns and regularities and is different from object-oriented or component-based methodology. What we are interested in is more at the lowest source-code level. While they are different, plagiarism does not contradict with either of these approaches as patterns, in principle, can be packed up to form modules. The design of classes or components for some applications is a mentally-intensive process. Reusable components by themselves do not demonstrate how they should be reused. It is the users' responsibility to decide how a component or a task or process can be repeated to achieve the results that the original designer, which may have different level or expertises or cultural background, has achieved. Reusability and repeatability are linked to pattern and people. These two concepts are intertwined, if not the same and they are key concepts adopted by the PbP approach we propose here.

There has not been much study on how much time to take, particularly for work-in-progress, programs by other developers and how this process can be controlled has not been addressed much. During the course of development, it is inevitable that we have to deal with the handing over of tasks due to departure of programmers, etc. How efficiently this can be performed depends to a large extent on proper documentation,

whether or not ex-developers are available for questions, and the experience and capability of the newcomer.

While computer programs could be very complicated and their logic difficult to understand, it should be noted that, if written according to PbP, these programs have to be made available for plagiarism without programmers having to spend time interpreting them. Written in the PbP way, the code that need to be modified, the types of changes that need to be made, the code that need to be kept unchanged, etc., have to be easily identified. In PbP, there are those who produce code to be plagiarized and those who plagiarise. The former adopts PbP because of the need to provide original or samples to be plagiarised by an inexperienced software team and the latter is involved because they are either not capable of producing similar work all on their own or they intend to minimize their workload. The way of this architecture optimises development costs.

Our principle and management vary from an old idea of a "chief programmer team" requiring skilled and experienced people to take responsibility of the complete software development process [1]. In PbP those who produces the originals may not need to be the chief programmers because they may not be involved in the implementation of user requirements and other processes in a software project. Those who revise the originals based on user requirements do not play the role of backup programmers or librarians, because they are now responsible for transferring user requirements into real pieces of program. Besides this, we remain sceptical about chief programmers having to be strong on managing inexperienced developers as people management involves excellent communications, interpersonal skills, leaderships, vision and a realistic appreciation of cultural issues. PbP is to de-emphasize the importance of addressing issues related to differences in people and environments but to stress quick replications of previous work.

In addition to being a tool for coding, PbP is of benefit to software process management. While effective management of software processes makes it easier for one to acquire a boarder picture of all activities within a software development organization, PbP embraces design and coding at a glance and its impact of coding on other processes such as integration, testing, maintenance, coding review, quality assurance and the costs can be significant.

The development of the PbP was inspired by pattern theory [2] and transfer of training in cognitive science [3]. PbP was developed to ensure the success of software development in a way that simple coding instructions need only be given even to inexperienced programmers. It was developed also to resolve different managerial and cultural issues relating to the working with inexperienced programmers. To describe PbP and the rationale behind its development, we first introduce a metric called *Inexperienced-Programmer Ratio* (IPR). IPR measures how much experience a software team has. With IPR defined, we then present the principle behind the development of the PbP and the details of the programming paradigm. We conducted a control experiment to prove the idea of PbP scientifically. The costs of software development are analysed afterwards. PbP has been found to be effective when tested with people from different background and culture.

Inexperienced Programmer Ratio

One characteristic of a good software process is that it can bring about the same effect repeatedly. Also, the same process is to be applicable when involving different people. Under an extreme circumstance, the process would be truly successful even when the members of a software team that adopts it consists of totally inexperienced programmers. In [4], Steve McConnell cited evidence that early object-oriented projects succeeded because they were staffed with bright people who could code brilliantly well in all languages. In fact, any design of any programming process, if adopted by programming talents, stands a very good chance of success. Contrarily, a software project might end up in disaster if it consists of a team of inexperienced programmers.

A software team is normally composed of a number of people with different levels of IT skills and knowledge. Some of them are more experienced and some are less so. While most IT managers prefer recruiting a team of very experienced members, it is, in many cases, impossible. It is often the case that experienced programmers with the right skill sets be unavailable. The ratio of experienced to inexperienced members in an IT team usually varies from region to region and from one company to another. Very often, it is also related to a company's development budget; that is, the company cannot afford to hire qualified professionals. In a nutshell, two factors determine such a ratio. One is external to a team. It is more difficult to find qualified programmers in less-developed than more well-developed regions. The other is internal to a team. It includes such factor as the expectations on IT and the financial situation. Fig 1 gives some possible scenario.

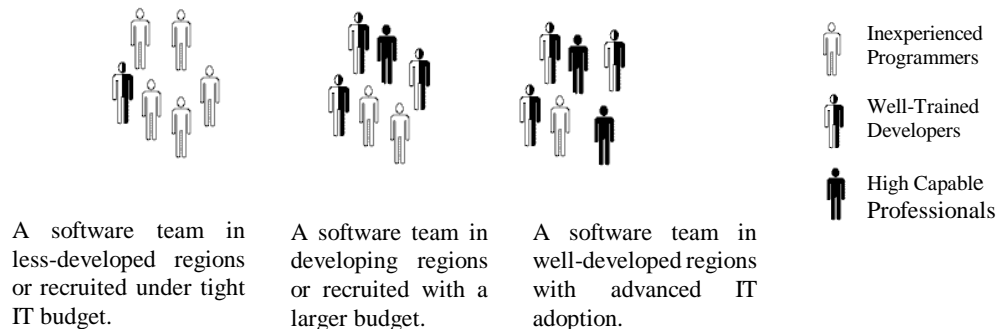


Fig 1: The Composition of Software Teams

Given the above, we define an inexperienced-programmer ratio (IPR), which is a metric to measure the experience of a team in terms of its percentage of inexperienced members, as follows:

$$IPR = \frac{\text{The Number of Inexperienced Team Members}}{\text{The Total Number of Team Members}} \times 100$$

An *inexperienced team* is a team with a mean IPR of over 80. According to [4], IPR can be an important factor determining the successes of a software process. A coding process which works well with an IPR of 80 should do so with 20 but not vice versa. This is particularly the case since the differences between two teams each of 80 and 20 IPR, respectively, are not only in the IT knowledge of the members but also the personalities.

In this connection, recent studies have been carried out to investigate into the cultural influences on software process improvement [5, 6]. IPR can be used as an indicator for a mechanism in which it was applied successfully. In software process improvement, IPR in nature is irrelevant to cultural impact whereas IPR is much more objective assessment to team management. Nonetheless, communication barriers, excluding language, between experienced and inexperienced people are definitely larger than ones among experienced programmers from different cultures.

Active rural industrialization attracts many investments in manufacturing business. Companies that set up plants in these areas are usually those that would like to exploit the low initial set up costs and to invest a minimal amount of money into its running operations. These companies are usually the ones that operate under tight budget control and they are not willing to spend much on IT. An economical, balanced IT solution that is imperfect but workable is all that they expect. It is noteworthy that the success of the investment in manufacturing industry there bets on a number of strategies: Sales, Marketing, Logistics, Finance and IT. Thus, a competitive information system may be a key to success, but not the most important one. As a consequence, software teams in these plants always consist of local people who are neither experienced nor well-trained in computing. These software teams are to support factory's operational applications. Compared with cost, Table 1 gives salary ranges of programmers in Huizhou (a less developed area in China) and Hong Kong (100 km far away from Huizhou). While the distance between HK and Huizhou is a 2 hours drive, one might argue that a team in combination with HK and local people could provide a best solution. However, top management who are an IT layman but a financial expert, always see another view in which we mentioned above.

<i>Year of Experience</i>	<i>Monthly Salary for Programmers in Less Developed Areas in China</i>	<i>Monthly Salary for Programmers in Hong Kong</i>
Student Plant Practice	<i>No Paid</i>	<i>N/A</i>
1 or below	<i>Less than Euro 157</i>	<i>More than Euro 1,500</i>
5 or above	<i>Less than Euro 400</i>	<i>More than Euro 4,000</i>

Table 1: Salary Range in Less and Well Developed Areas

Plagiarism Based Programming (PbP)

In this section, we give the details of the PbP approach in which we had implemented in a less-developed region in China. According to [2], the pattern theory could be considered as an approach to arrange workspaces so that new employees would be able to learn by being in proximity to their mentors. This is called "Master and Apprentices." While this aspect of the pattern theory is useful for the problem being considered, it should be realized that some patterns might be difficult to be understood and followed. For PbP, it is important that the resulting programs should be easy-to-follow for the purpose of plagiarism.

Fig 2 illustrates our main idea. Path 1 indicates that an original program is written to handle one particular task. By modifying part of the source, such as converting a for-loop into a while-loop, the program still achieves the same task (i.e. path 2). The

modification, of course, is quite common among Computer Science students who try to complete a programming assignment in the last minute by plagiarism. In some cases, it should be noted that plagiarism could be used to solve a task similar to the one the program originally intends to solve. Thus, a piece of code revised based on the original program may perform a similar yet different task (see path 3 in Fig. 2). For PbP, our concern is how to write the source or design patterns so that it can be used as the original to facilitate plagiarism.

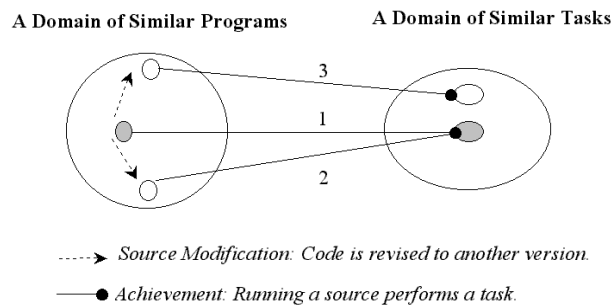


Fig 2: Illustration of Programming by Plagiarism

In [7] and [8], we introduced MDC (Managing Design-Coding) for the design of patterns. A generic application-independent architecture could be built as a set of patterns. An extension of the MDC is to take easy-to-follow for plagiarism into consideration. To do so, we highlight each pattern in three different colours. Blue colour is used to represent no-change; green colour is used to indicate that the code concerned requires reading; and red colour is used for the part of the code that requires modification. A piece of code is therefore like a series of colours of the form [B G R B G B R B ... B]. The colouring makes plagiarism possible in real, competitive business environments. The purpose is to make sure that copiers know where to pay attention to and where changes are required to make.

As an illustration of PbP, we consider a delete operation. Fig 3 describes a piece of code that is for a row deletion in a database. The code includes logical deletion (updating a flag column), transaction rollback and a mechanism for error checking. For plagiarism, we are concerned with 3 pieces of semantic information:

- (1) name of table where a row is to be deleted
- (2) condition under which a row is to be deleted
- (3) number of rows in which the operation affects (by default, only one row of deletion is allowed)

For example, when a programmer wants to delete a unique invoice INV1MAY01, what he needs to do is to copy Fig 3 and replace

```
tablename with INVOICE
col='condition' with INVOICE_NBR=' INV1MAY01'
```

```
Update tablename set status = 'D' where col='condition'
if @@rowcount<>1 // number of rows affected in this operation
                // if more than one row is affected, set
                // @@rowcount<=1. If only 2 rows are affected,
```

```

// set @@rowcount<>2
and @@error<>0
begin
  raiseerror 50000 'Error in update tablename set status'
  exec master..xp_logevent 50000
  rollback transaction
  return
end

```

Note: Bold text represents blue (neither read nor write)
 Italic and lighted grey text represents green (read only and write occasionally)
 Normal text represents red (read and write)

Fig 3: A sample code for database deletion. When the above is highlighted with colors, we can tell, by visualization alone, how much to change for this piece of code.

For a team with 80 IPR, the programmers read the green part of the source and read/write the red part. Blue part is copied exactly. Many technical problems covered are transparent to them. Inexperienced programmers work with fewer difficulties as they are all performing plagiarism. What they need to revise is confined to the blue and green parts. It is noted that, for the object-oriented approach or the component-based methodology, the internal of an object is blue and the interface is red.

Experiment

We conducted a control experiment to investigate the underlying idea of PbP that plagiarism *alone* helps inexperienced programmers. In order to prove it workable objectively, we eliminated colouring in original by PbP and hence minimized the probability that some subjects could achieve the assignments by chance. Like many cognitive experiments, we provided no spoken instruction to them. It is noted that the experiment is much stricter than the working environment of PbP (or even any other software development) in which verbal and informal communications provide a great deal of feedback and help.

The evaluation involved 16 programmers working in a less-developed region in China. The experiment was designed to have two parts. Part I would be held before Part II. For Part I, the subjects were required to write a program for computing n factorial ($n!$) in any language or even in pseudo code. Since the programmers were inexperienced, many actually found it difficult to write a factorial program without any referential material. After they had submitted their answers for Part I, they then proceed to Part II. In this part, we provided three complete SQL programs of 3^n written in different ways: if-then, while loop and recursion, respectively. The programmers were requested to revise each of them for the calculation of n factorial. In addition of being semantically correct, those programs have to be machine compliable; that is of no syntactic error. In the end of the experiment, each person would deliver 4 pieces of source.

a program of n factorial written by the subject alone

Part I

:

three programs of n factorial based on another three programs of

Part II 3ⁿ using the if-then, while loop and recursion techniques,
: respectively

The purpose of Part I was to examine the ability of subjects in justification that they may excel the task immaterial to contributions of plagiarism. Thus the sequence of Part I and II could not be reverted. We were particularly interested in those who failed in Part I but passed Part II. This indicated that plagiarism aided inexperienced programmers. Although Part II seemed to contain three assignments, they were independent in terms of logic. In this case, we should compare Part I with three results of Part II, respectively. Based on a programmer passing or failing the two parts, there would be four possible outcomes for the experiment (see Table 2).

	Part I	Part II	Comments
	✓	✓	No conclusion! Subjects could be capable of the given assignments
	✓	✗	Very Unfavourable to Plagiarism
	✗	✗	Unfavourable to Plagiarism
	✗	✓	Favourable to Plagiarism

Table 2: A range of four results in the experiment

Point 1 of Table 2 shows that a subject can manage to complete both. No conclusion is reached as the subject could do Part I by himself. Point 2 will overthrow our belief about plagiarism if people are able to complete the assignment alone but fail to do it in part II. Point 3 is unfavourable to plagiarism, as it demonstrates a fruitless attempt for Part II even though it is possible that a subject is not knowledgeable about programming at all. Among these outcomes, only the fourth is favourable to plagiarism. In our experiment, the results fell in point 1, 3 and 4. No case about Point II was ever obtained.

We provide a detailed result done by one of the subjects in Table 3. Obviously, the subject did not know how to write a program of n factorial. Moreover, the person had never done programming in SQL. Part II contained three sample programs to be plagiarised. Interestingly, he was now able to modify two SQL programs correctly, in Part IIa and IIb of Table 3. Part IIc was the most difficult one as the sample program involved a recursive technique that would make plagiarising hard for some people. As discussed earlier, some patterns were harder to follow. It is concluded that the original programs of 3ⁿ by if-then and do-loop have a higher value in terms of reusability and repeatability than the one by recursion.

Part I	Part II a	Part II b	Part II c
No Plagiarism	Plagiarising an IF-Then Program	Plagiarising a Do-Loop Program	Plagiarising a Recursion Program
✗	✓	✓	✗
A=1 B=1 C=1 Input N For N to 1	Create proc factorial (@y integer) as begin if @y = 0 return 1 if @y = 1 return 1	create proc factorial (@y integer) as begin declare @result integer select @result = 1	create proc factorial (@y integer) as begin declare @result integer declare @y_minus integer if @y = 0 return 1

<pre>A+1=A A*B=C C=B End For Output C</pre>	<pre>if @y = 2 return 2 if @y = 3 return 6 if @y = 4 return 24 if @y >= 5 print "out of the range!" end</pre>	<pre>while (@y >=1) begin select @result = @result * @y select @y = @y -1 end return @result end</pre>	<pre>select @y_minus = @y - 1 exec @result = fatr @y_minus if @y = 0 return 1 select @y_minus_1 = @y_minus - 1 exec @result = fatr @y_minus_1 if @y = 0 return 2 select @y_minus_1 = @y_minus - 1 exec @result = fatr @y_minus_1 if @y = 0 return 6 select @y_minus_1 = @y_minus - 1 exec @result = fatr @y_minus_1 if @y = 0 return 24 select @y_minus_1 = @y_minus - 1 exec @result = fatr @y_minus return (3 * @result) end</pre>
---------------------------------------------	----------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 3: A Detailed Result of a Subject who had no knowledge in SQL programming

Fig 4 summarizes the outcome of the experiment. In Part I, only 4 out of 16 (25%) succeeded in coding the problem; while 10 of 16 (62.5%) were able to complete the assignments in Part II a & II b, and 7 of 16 (43.7%) in Part II c. It is noteworthy that pass criteria for Part I and Part II were different. The previous was simply justified by the expression of logic while another was demanded to be executable. Comparing Part I with Part II, i.e. (I vs. II a), (I vs. II b) and (I vs. II c) , the percentage of the number of success grew in 150%, 150% and 75%, respectively. From cognitive science's point of view, our experiment also contributes towards the understanding of systematically-transfer and transparency-transfer in human mind [3]. Fig 4 confirms that plagiarism provides certain degree of assistance in programming by inexperienced people.

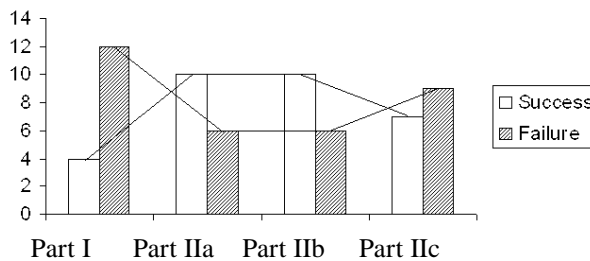


Fig 4: An Overview Result of Our Experiment

SPI and Cost Reduction

PbP has the advantages that, even with a high-IPR software team, software process improvement and cost reduction can still be achieved. PbP makes it possible for successes of software projects repeatable; theoretically, it is able to repeat the success in any low or high IPR team. Directly or indirectly, this also reduces costs. When we work on projects of similar nature, say a database application, the adoption of the PbP is expected to lower costs as teams with relatively high IPR can be built. In [7] and [8], we already introduced a technique involving the use of basic patterns such as insert, update, transaction and error control in the design of a set of templates. These templates can be used as originals for PbP. The design and use of these templates has a number of implications on SPI.

Fig. 5 describes a traditional software development lifecycle that is structured as a set of sequential processes. A process started earlier usually has some impact on those started later. Thus, the successful completion of a process depends both on itself and on others. If a process is managed well, therefore, other processes also benefits. This naturally results in cost-reduction. PbP can be adopted in both a design and a coding process. Because of accumulated effects, the adoption of PbP also improves other processes such as testing, integration, and maintenance, etc.

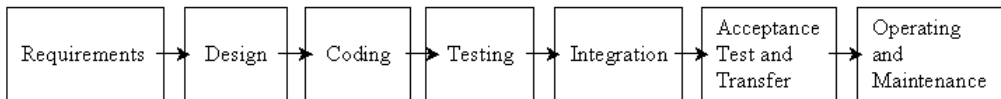


Fig 5: General Software Development

When developing a system, processes will be adopted to ensure that the system is robust in logic and that it meets the requirement. To build a reliable system, it has to be free of technical problems, such as data integrity, locking in multi-user mode, security and it is to be able to handle expected error gracefully, etc. To meet requirements, it should be noted that PbP standardises coding style and pattern for coding. Based on the PbP, when code of original is properly highlighted in blue and green colour, the robustness of code will be passed along for the meeting of systems with similar requirements.

Based on our experience, we observed that robustness of code can be achieved through coding pattern. This helps avoid, without special managerial control, (un)intentional human misbehaviour for inexperienced people such as forgetting to handle return code of a transaction or being too personally confident about his own code to eliminate unnecessary error-checking. As a consequence, workload for later processes such as those related to testing, integration, and maintenance is reduced, so are costs. With lessened workload relating to technical skills of building a reliable system, more focus can be put on the requirements.

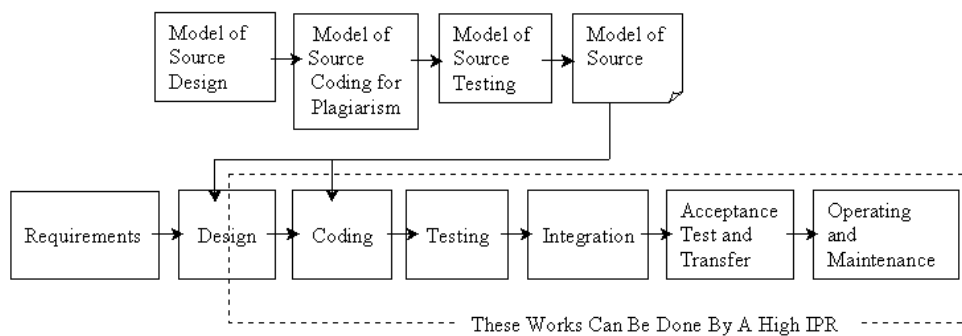


Fig 6: Software Development Using Programming by Plagiarism

Fig 6 illustrates software development by plagiarism. Model of Source Design, Coding and Testing altogether attempt to build a set of patterns. For example of database application, we have patterns of inserting data into, deleting it from and updating it of tables which are irrelevant to functionality of the application. Model of Source, which is

used as originals highlighted in three colours for plagiarism, corresponds to Path 1 of Fig 2. Given new requirements, design and coding based on Model of Source is then proceeded with. In [7] and [8], design is highly related to coding implementation. A design process now merges requirement analysis with Model of Source. For example, if deletion of Model of Source is written in a way of marking an invalid flag of a row of a table as deleted rather than performing physical erase, the database of our application has to be designed in the same way; that is, logical deletion replaces the function of a physical deletion. Whether we employ physical or logical deletion depends on Model of Source. Here the emphasis is put on the relationship of design to that of coding.

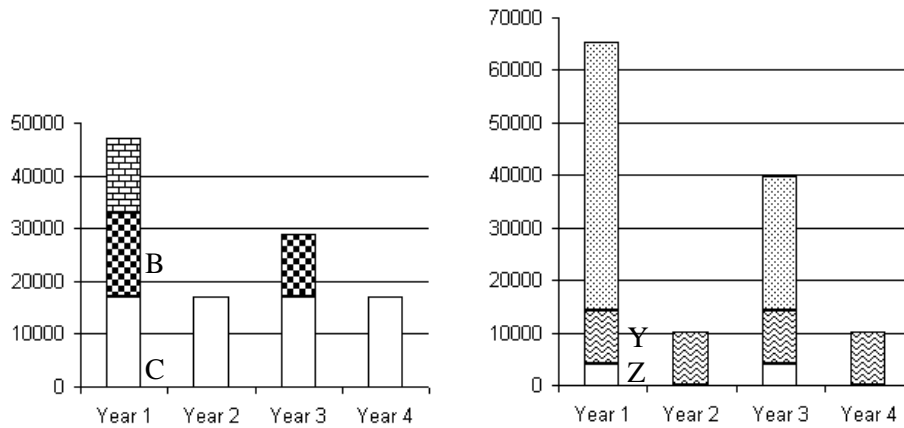
(Physical Deletion)	Delete an invoice AD135 from a database
(Logical Deletion)	Mark an invoice AD135 as invalid in a database

Based on requirements and design, a sample from the source model is selected. We work on the modification of its red part. Of course, we also need to design a screen layout and program simple controls such as closing a window and printing contents. Starting from coding, all work later can be done by a high-IPR team.

For the testing processes, we will concentrate on verifying if application requirements have been met. Whether there are some error related to implementation of the product based on the requirements such as calculation of a PO discount for major account customers can be easily identified, while we care less for whether the PO database gets integrity problems due to code robustness.

When new requirements come in, there are two possibilities: minor or major source change. For minor, we probably revise those red colour parts for changes. For major, we need to re-work or extend an original design model. This will normally consume more resources. Fortunately, a high IPR team that understands application requirements may complete all these. As to IT managers, they are not afraid of the loss of programming knowledge due to personnel turnover in their software team. Due to colouring standardisation, a high IPR team will be able to performance the same job as those original developers were. This feature makes maintenance cheaper and easier.

For reference, we review costs of two approaches taken in breweries located in Huizhou (Southern China) and in Shanghai (Northern China), respectively. The former was to develop an in-house application by inexperienced programmers using PbP. The latter was to implement a European-based ERP package. While avoiding confusion, we could only count on the most obvious tangible cost: money; even though time, effort and skill composed costs. Both approaches were judged as successful in terms of user satisfaction, budget plan and project schedule.



Huizhou In-house Software Development By PbP Shanghai Third-party Software Package

- A: Two Months Salary of Two Qualified Programmers for Original Patterns Development for Plagiarism
- B: Development tools such as PowerBuilder and Microsoft SQL Server.
Note that the tools needs to be upgraded in latest version every two years
- C: One Year Salary of Four Inexperienced Developers
Note that the pay remains constant along the time due to personnel turnover
- X: Third-Party Software Package. Note that new release comes out every two years
- Y: Annual Maintenance such as hot-line support and patch
- Z: On-site Consultant Service

Fig 7: Cost Overview of Two Software Projects

The total amounts of Huizhou and Shanghai Information Systems for four years are 110,000, and 125,000 in Euro respectively. However, the in-house system developed by Huizhou was further revised for local requirements and implemented in four other regional offices, which significantly reduced costs. Normally, the cost of developing an Enterprise-wide in-house system for one site is higher than implementing a third-vendor product due to the salary of programmers, particularly if we project the running costs over 4 or 5 years.

Conclusions

It is often arguable whether a software methodology will work as well as its predecessor. If those implementations indicate an IPR, we will be able to evaluate the repeatability of the application in other teams in other countries in a much more scientific way.

Most study is interested in straightforward improvements such as in terms of time and money shown in the vertical line of Fig 7. We pay an attention to one more dimension people (measured by IPR) in a sloped line of Fig 7, however. A real successful software project nowadays has to take the consideration of people and costs in a variety of environments. This will best fit in the 21st Century Software Development scenario where software development will be distributed in both developed and developing world.

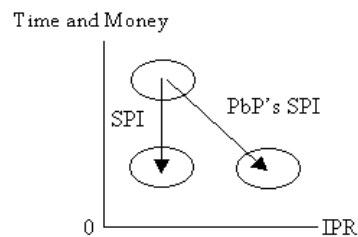


Fig 7: Current Study of SPI vs. PbP's SPI

People management is a complicated topic as some unchangeable issues on culture and human cognition are ill-defined in engineering discipline. However, plagiarism-based programming attempts to resolve these uncontrollable problems about human and external environment. We conducted an experiment in order to observe the connection between plagiarism and programmers. The result confirmed the application of Plagiarism-based Programming.

Finally, we would like to thank Jørn Johansen and Karlheinz Kautz for their helpful comments.

References

- [1] Sommerville I, *Software Engineering*, Fifth Edition, Addison-Wesley, 1995
- [2] Coplien JO, The Origins of Pattern Theory: The Future of The Theory, And the Generation of a Living World, *IEEE Software*, pp. 71-82, September/October 1999
- [3] Schumacher RM and Gentner D, Transfer of Training as Analogical Mapping, *IEEE Transactions on System, Man and Cybernetics*, pp 592-601, July/August 1988
- [4] MsConnell S, *Rapid Development*, Microsoft Press, Redmond, Washington, 1996
- [5] Siakas K and Balstrup B, A field-study of Cultural Influences on Software Process Improvement in a Global Organisation, *Proceedings of European Software Process Improvement 2000*, Denmark, November 2000
- [6] Sztaki BM, Messnarz R, and Sandberg J, The perception of Quality Based on Different Cultural Value Systems, *Proceedings of European Software Process Improvement 2000*, Denmark, November 2000
- [7] Lui KM and Chan KCC, Managing Inexperienced Programmers by Managing Design-Coding, *Proceedings of European Software Process Improvement 2000*, Denmark, November 2000
- [8] Lui KM and Chan KCC, Managing Design-Coding for Software Development in China, *Proceedings of Software Engineering Application*, 2000, USA

Session 2 - SPI Experience Stories

**Session Chair:
Nils Brede Moe, Sintef, Norway**

Life on Level 5	2-2
Irish Experiences with Software Process Improvement	2-11
Process Psyops - Moving a Large Well Established Company Toward SPI	2-21

Life on Level 5

Bart van der Wal
Atos Origin, Eindhoven

With help of:
Darayus Desai
Atos Origin India

Introduction

This is the story of Atos Origin India; a story of a company reaching level 5; a story also of enthusiastic software engineers and enthusiastic management. So most of all it's a story about people; but it is also the story of several plans, attempts, training, failures, assessments etc. and it all started about ten years ago.

And this story still continues.

For me it started January 1996 when I was invited for an assessment. My CMM and Software Process Improvement experience till that moment was with level one and less companies. Within Origin I had implemented ISO 9001 Quality Systems and as a project manager I thought I had worked on level 2.

Never before I had been in India. I did not know anything about the culture, about the people. I thought India was a 'low wages' country where cheap quick and dirty programming was the standard

Atos Origin India

Atos Origin India is the Indian daughter of Atos Origin, a worldwide operating ICT service provider.

Atos Origin India is located in Mumbai and provides software-engineering services mainly to clients in Europe and the United States. The types of services they deliver include software engineering, ERP package implementation and customisation. The technical staff consists of about 400 engineers. The company is organised in Service Practices, each focusing on a specific product or business area. At the end of 2000 there were six Service Practices.

Within these Service Practices the service delivery is based upon the concept of Production Lines. A Production Line is very client oriented and is set-up by defining dedicated processes, people and infrastructure based on the requirements of the Client. Specific Production Line measures are defined related to the client's objectives and used to control the processes and its output at each stage and to check quality at different

points in the Production Line.

The concept of setting up Production Lines is the logical consequence of the strategy to create a long-term and mutually beneficial partnership between Atos Origin India and the Client. The Organization Standard Software Process (OSSP) is the basis of the processes defined for the Production Line. These OSSP's are tailored when necessary to meet the specific needs of the Production Line.

Each Service Practice has a number of Production Lines catering to diverse Clients. For implementing and checking the performance of the processes the so-called ProChamps (Process Champions) are responsible. Every Production Line has at least one ProChamp. Together with the SQA function per Production Line they are responsible for the Software Quality Assurance and process improvement within the Production Line.

How did they reach level 5?

It all started in 1993 when Origin India installed an ISO 9001 project. After the implementation of the ISO 9001 Quality System in 1994, the next goal was CMM level 2. But why ISO 9001, why CMM level 2? This is not a goal in itself!

The vision of Atos Origin India was clear and visualized in their so-called Quality Equation: product quality + process quality + service quality + value addition fused together with dynamic personal interaction will ultimately result in Customised Quality or better in Customer Delight. The goal was to become competitive not only in terms of price, but also in terms of schedule, anticipating and fulfilling client needs, overwhelming services, reliability and defect free packages.

The first CMM assessment was executed in 1996 and I could not believe my eyes. After two days of the Assessment I got the feeling they were fooling me, that they had learned their lessons well. It was weekend and I changed my plans. No sight seeing in Mumbai. No I stayed in the hotel and checked all my notes, all my findings. And then I changed my assessment schedule. I decided to use the whole Monday for assessing a project that was not on my schedule in the beginning. I took them by surprise. But they survived. There were no 'learned lessons', it was daily practice, and they were really very good. The conclusion of the assessment was that Origin India had almost completed level 2 and level 3 of the Capability Maturity Model. Improvement plans were advised and implemented to reach level 2/3. Four years later, level 5 was attained for 3 of the 6 Service Practices.

What happened in those 4 years? An enormous effort within the whole organisation to improve and to improve and to improve. The commitment of the management was great, the enthusiasm and ambition of the software engineers, even greater.

In 1998 I had the privilege to execute a Quick Assessment and could tell the people that level 3 was almost fulfilled and that level 4 was very close.

Related to my experience in the Netherlands, Belgium and UK, Origin India was at the top. Yet, their ambition was to reach level 5, because they strongly believe that the race for Quality has no finish line.

The first level 5 assessment was conducted at the end of 1999. In the summer of 1999 I had visited Origin India three times for execution of quick assessments, to check the progress of organisation-wide improvement actions and discuss this progress.

For the assessment we used the Process Professional Assessment Method of Compita.

The Software Engineering Institute has certified this PP Method. A Process Professional Qualified Lead Assessor led the assessment team. Three Service Practices were within the scope of the assessment. The other three were not so far, imagine they were already level 3! The outcome of the assessment was that level 5 was a step too big, but the 3 Service Practices reached level 4. At that moment the organisation realised that they needed a fundamental change in their approach. The only way to reach level 5. Not an approach per Service Practice but a joint effort. So the best practices of all the Service Practices were shared, and a concerted effort was put in to facilitate sharing of knowledge between the Service Practices. Process Action Teams were implemented with members of all the Service Practices and a strong Management participation. And all the KPA's of level 5 were combined in an overall approach of Process Improvement

What does level 5 mean?

Living in a level 5 organisation means working in a very professional environment, with very professional colleagues. It means that professionalism, a strong Client focus and the continuous drive for learning and improving is natural behaviour. For the daily work, the right tools are available and the procedures are not bureaucratic, but well fitted for the job. People understand the processes and their purpose.

It means that there is no wasted time due to missing documents, missing information and ill-fitting processes. It means that the real creativity is focussed on finding the right solutions for the Client.

It means always exchanging experiences, learning from mistakes as well as strengths and implementing the best practices based on these experiences.

In CMM terms:

For the first levels of the CMM, there is no discussion anymore, about the basic principles of project management (level 2) and the organisation level issues at level 3. That is 'standard way of working, intrinsic behaviour, a basic fact of life.

Atos Origin has defined an Improvement Management Process, which makes it possible for every employee to raise an Improvement Management Suggestion (IMS). These IMS are controlled and tracked by the ProChamp of the Service Practice. But the Improvement Management Process is not only the instrument for suggesting Improvements, it's also the body of the whole learning and improving system of the organisation. The Processes & Quality (P&Q) function is responsible for Improvement Management at organisation level. Besides employee feedback, inputs for improvement are obtained from a host of other sources like Client feedback, Internal and External Quality Audits, Quantitative Process Measurements and Defect Prevention activities.

For major changes so-called Process Action Teams (PAT) are installed, during the lifecycle of the improvement. In these PATs, people from all levels of the organisation are involved.

Before implementation the effects of the change are piloted. Their results in terms of effectivity and efficiency are evaluated and compared with the expected results. There will be no implementation when these results are negative.

Benefits and costs or results

The benefits of continuous Software Process Improvement for the last ten years are clear. So are the costs. When you ask the Managing Director of Atos Origin India what the costs of SPI have been, his answer is always “nothing, it has given us only benefits.”

Atos Origin India spends around 3% of its organisational effort annually (3.3% during the year 2000), on SPI activities. This includes a core team of four full-time persons, along with several persons working part-time on various improvement projects, as and when they are initiated.

While Atos Origin India has realized returns in various areas, one of the major benefits has been the constant upward trend in client satisfaction. While no client gave a rating of 5 (on a scale of 1 to 5) until 1997, today 40% of the client satisfaction surveys have a rating of 5! And they have built strong long-term partnerships with their clients. They have also seen a significant improvement in the defect-free deliveries and achievement of planned schedules.

The bottom line has also shown a strong positive trend over these years, although there were various other factors too, besides process improvement, which contributed to that. While it is not possible to establish a direct relationship between process improvement and the improved profitability, it can certainly be said that the ROI is significant, in terms of improved client satisfaction, predictability, productivity and profitability

Key Factors of success

Atos Origin India identified a number of key factors that have contributed to the success of the company.

The most important ones are:

1. A very strong management commitment
2. Very professional software engineers
3. Client orientation from the start
4. The Balanced Score Card as basic concept for the management of company
5. Improvement Management as overall concept
6. The ProChamp
7. The use of automated tools

Management Commitment

Within Atos Origin we have developed the Management Maturity Model. As the picture below shows the model knows four levels:

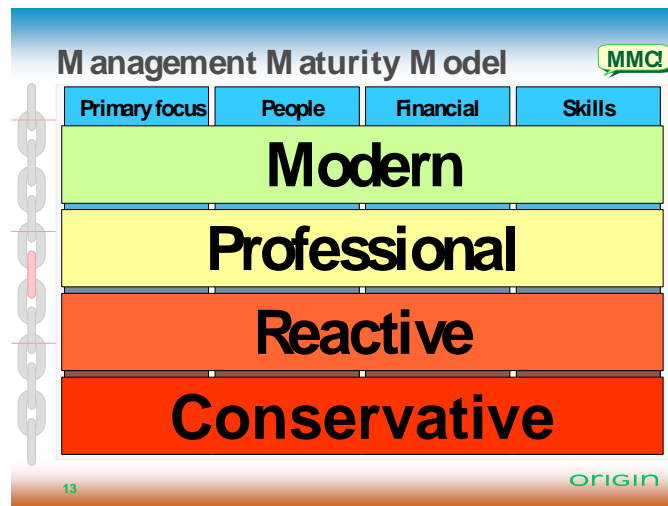


Figure BWL.1 : The Management Maturity Model

Conservative Management is primarily focused on effort; people are rewarded based upon their seniority and their credentials. The budgets are the most important performance indicator. **Reactive Management** is primarily focused on output; being a star is the most important attitude for the people; the costs are the KPI and skills are improved via courses.

The **Professional Management** is focused on the process and is strongly stimulating the working force to work in teams. Short-term investments are their financial focus and the people are trained to become professionals. At the highest level, the **Modern Management**, we see a focus on improvement, culture, long term investments, and learning. We strongly believe that only Professional and Modern Management is the right basis for a successful Process Improvement project.

Looking at Atos Origin we can see a shift in the Maturity of the Management during the last two years. Their primary focus shifted from process to improvement, from teams to culture; and to structural learning within the whole organisation. And it was this shift that helped them to reach level 5.

Senior Management is strongly committed now, but also from the start. They are the first players on the barricades and lead by example. They are steering SPI on the highest level, are involved in the Process Action Teams and are showing in practice their belief in SPI. All members of the Management Team were involved in the development of the Quality System.

Professional Software Engineers

The software engineers are highly educated, disciplined, very ambitious, they know all everything about their own job and the job of their colleagues. SPI is a real part of their lives.

They are strongly committed to the Client and above all they enjoy their work. Human Resource Management is very well organised. With every software engineer a Personal Development Plan is agreed at the beginning of the year. This plan contains agreements about the career, the training and the performance of the employee. The employee and his manager are committed both to realise this plan. At least twice a year the plan is

evaluated.

The results are clear. Every year an employee satisfaction survey is executed. The last five years the results of these surveys were better every year, from 4.0 to 4.6. Elements of the survey are job satisfaction, training, and career possibilities and management support.

Client Orientation

The Quality System is built to meet the needs of the business. This includes a very clear focus on the needs of the Client.

At the start of a Production Line Performance Indicators (KPI's) are defined and agreed with the Client. These KPI's are used to assess the product quality and process performance on a continuous basis. If the KPI's are not realised corrective actions are taken and areas of improvement are defined. Examples of these Performance Indicators are Timely Delivery, Defect-free delivery, Understanding of the Requirements, and Cost – Quality Relationship.

Balanced Score Card

The Balanced Score Card is the basic concept for managing the company. Based on the Balanced Score Card on company level, Balanced Score Cards are defined for each Service Practice and Support Group. General elements of every BSC are Financial, Customer, Operations, and Innovation/People Indicators. There is a direct relationship between the Company goals, the goals of the Service Practice and the Indicators on the BSC.

Per Indicator targets are set at the beginning of each year. The actual status per Indicator is visible at every moment.

Each Production Line has a Contract Dashboard, which is a tool for managing the operation of the Production Line. The Contract Dashboard is set up on the basis of the KPI's defined by the Client and Atos Origin India's goals (as defined in the Balanced Score Card). It is used to collate all the Production Line metrics and provide a graphical representation for each of the KPI's, enabling the Production Line Manager to manage the product quality and process performance.

Improvement Management

After the assessment in 1999, where level 4 was reached, there was a fundamental change in Software Process Improvement (SPI) approach.

The SPI activities are carried out under the direction and guidance of the SEPC (Software Engineering Process Council), which comprises of all Service Practice Managers, two representatives from P&Q and the Managing Director. The SEPC is primarily responsible for steering and overall management of SPI activities across Atos Origin India. The improvement opportunities given in the CMM Assessment Report were studied by the SEPC and grouped based on the key process areas of level 4 and 5. Seven PATs (Process Action Teams) were formed to address these groups of improvement opportunities. Each PAT comprised of members selected from different Service Practices, to ensure that each Service Practice was

represented and also that the information would flow back to all Service Practices. A PAT Leader, who was also a SEPC member, led each PAT.

The first step was to define a **process for the PATs** themselves; this was a task for the PCM-PAT, who defined the process, which was then reviewed by the SEPC.

Subsequently, each of the PATs followed the process, viz. making an Improvement Plan, studying the various improvement opportunities in light of existing processes, best practices within the Service Practices. They evaluated various alternatives, selected the appropriate one and enhanced the existing process or defined a new process, as necessary. This was again discussed and reviewed with the SEPC and few members from other PATs. During this review, pilot candidates were also identified for piloting various processes.

An interim meeting “SPI 2000” was held to inform people about the SPI activities and provide an overview of the new/enhanced processes in various areas. The piloting process was also explained.

Thereafter, the **pilots** were started; a Pilot Plan was made for each pilot candidate Production Line, describing the activities, resources, milestones, deliverables, effort and schedule. The Pilot Plan was implemented under supervision and guidance of PATs. The pilot progress was reviewed periodically by the SEPC. The pilot results (including problems, lessons learned, benefits, etc.) were then evaluated by the SEPC and the final decision made (institutionalise, re-plan or terminate).

*The **Improvement Management Process** was also enhanced by the PCM-PAT, to clearly segregate reactive and proactive improvements and add more details for each. Details about the PAT process, piloting process and institutionalisation have also been added. A Goal-Process Matrix was also prepared to identify key processes for structured or for proactive improvement. The diagram below describes the various sources or inputs for the improvement management process.*

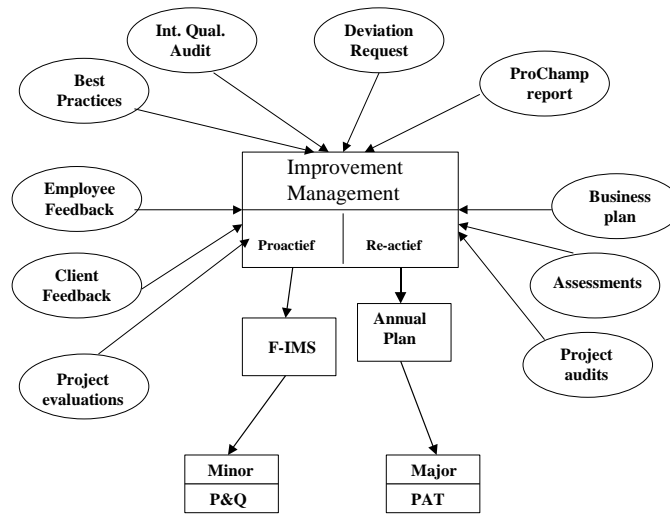


Figure BWL.2 : The Improvement Management Process

Improvement Management is treated now as a total concept. The related level 4 and level 5 KPA's, like Technology Change Management and Process Change Management, are integrated in one way of continuous learning and improving.

The ProChamp

The implementation of the ProChamp has given Atos Origin India extra speed in reaching level 4.

In the daily practice it proves to be an effective vehicle for creating a quality culture, promoting sharing of knowledge and experiences, implementing quality assurance and process improvement activities within the Service Practices, and across the whole organisation

Automated tools

Every Production Line makes extensive use of automated tools. These tools cover workflow, time recording, reviewing, measurement and process capability.

On organisational level everything comes together with tooling for the overall process database. Parts of the development process are structured and managed as real workflow management with triggers etc. and automated recording of measurements.

Conclusion

What can we learn from Atos Origin India?

Software Process Improvement takes time, it will cost effort and money. Software

Process Improvement needs a Long Term Vision. But at the end the benefits are clear. The quality of all the processes is improved, also the quality of the end products and the quality of the services, which you can see for example in improved effort variance, improved rework variance.

Client satisfaction is improved and also the Employee satisfaction. All the goals, defined at the start of this journey are achieved!

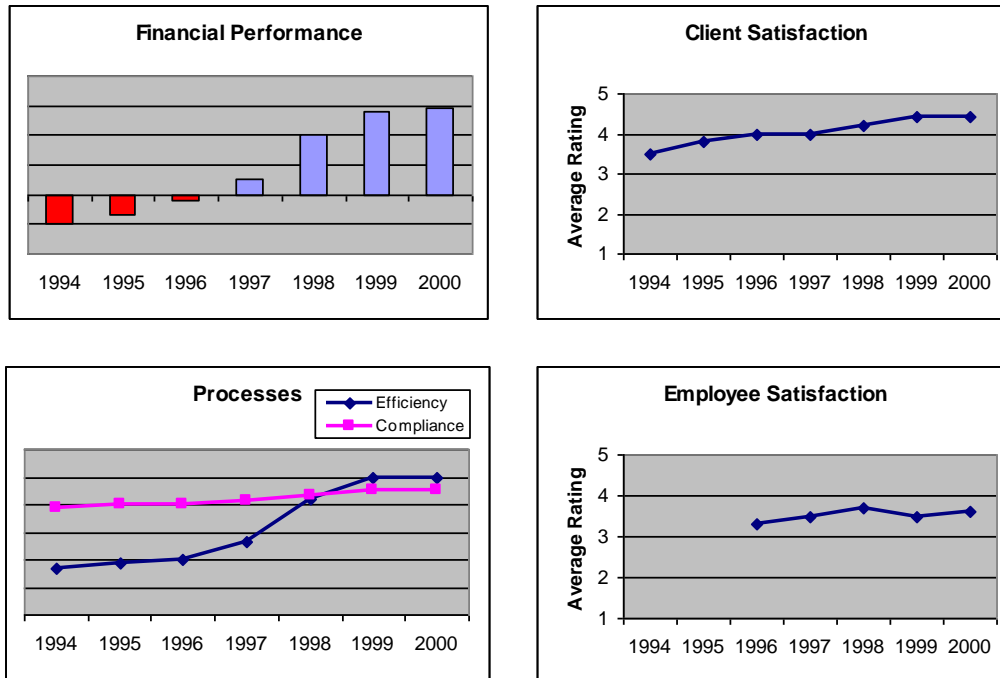


Figure BWL 3 : Performance Indicators

Why is Atos Origin India successful? In my opinion the Key success factors are clear: It is a strong Management commitment, visible every day in all his aspects. And there is a very disciplined working force with a strong ambition. But it is also the Maturity of the Management. A very mature management is perhaps the most important condition for a successful journey to quality. Is this the end? No, of course not. They have come a long way, and they still have a long way to go. As Darayus Desai, the P&Q manager says: “The race for Quality has no finish line.”

Irish Experiences with Software Process Improvement

Fran O'Hara
Insight Consulting Ltd.

114 Granitefield
Dun Laoghaire
Co. Dublin, Ireland
Ph: +353-1-2854510
fran.ohara@insight.ie
<http://www.insight.ie>

ABSTRACT

This paper/presentation will provide a brief perspective on Software Process Improvement (SPI) in Europe – its history, current situation and future direction.

Four case studies will then be presented covering a diverse range of business domains, organisational sizes and approaches to SPI. The objective is to share key lessons learnt from these diverse experiences.

The author has worked closely with each of the organisations involved in support of their SPI programmes. The case studies will show the starting position of each company, the approach taken, results achieved and lessons learnt. A number of themes such as assessment approach used, cultural/people issues, etc. will be used to explore the experiences of the various companies. The four case studies are

- NewWorld Commerce (formerly Cunav Technologies),
- Motorola Cork,
- Silicon and Software Systems and
- Allied Irish Bank.

Assessment models used include SPICE (ISO/IEC TR 15504) [1] and Software Engineering Institute's CMM¹ [2] (one organisation also achieved ISO9001 certification).

¹ Capability Maturity Model is a service mark of Carnegie Mellon University and CMM is registered in the U.S. patent and trademark office.

Keywords

Software Process Improvement, CMM, SPICE

SPI – the European Perspective

In Europe there has been quite a significant interest in and uptake of Software Process Improvement. Historically, the quality management system approach using ISO9001 certification was the approach of choice for many companies in Europe. Schemes such as the UK TickIT scheme provided additional software emphasis to the certification. However, there was a growing interest in incremental software process improvement based at least in part on the significant uptake of the Capability Maturity Model [2] in the U.S.

Unlike the U.S. strategy of providing funding for the development of tools to support SPI (i.e. the CMM and associated deliverables from the SEI), the European Commission chose a different strategy to stimulate and support the adoption of SPI. They provided funding directly to companies under the European Software Systems Initiative (ESSI) scheme. ESSI funded many programmes including, for example,

- awareness and training actions,
- direct funding to over 200 SMEs (small and medium enterprises) to support Process Improvement Experiments (PIEs – see [3] for the PIE case study repository).
- dissemination actions

The focus was on subsidising organisations to adopt best practices with the hope that this would stimulate further improvements in those organisations and more widely in the software industry. However, the current status of SPI in Europe is that although the ESSI initiative achieved a great deal and SPI is coming more into the mainstream, the breadth of uptake of SPI across organisations is still somewhat fragmented. Local content at European SPI conferences (e.g. European SEPG, EuroSPI) is now of a very high quality but SPI has not yet become ingrained into the culture of the software industry in the manner that it now is in the U.S.

Awareness of the SEI CMM is however high in Europe and there are many organisations who use it as a ‘toolbox’ for improvement rather than using it as the basis for a formal process improvement programme with the associated formal CMM assessment. Indeed, there are a number of factors that may indicate an increasing widespread adoption of CMM...

- availability of experienced SPI personnel from larger CMM-based organisations moving to other organisations and facilitating effective practical improvements using the CMM
- increasing availability of CMM lead assessors (both in the SEI’s assessment method, CBA-IPI, and also in the SEI-accredited assessment method from Compita Ltd., PPA for CMM)
- more experience and data on benefits achieved from CMM improvement programmes resulting in a market driven CMM emphasis based on expected competitive advantage on one hand and it being a required supplier certification on the other

SPICE (15504) is now an ISO standard and should be of interest to those who want to either focus on a few processes to improve or those who want to widen the scope of their

improvement program beyond software development (which is the focus of CMM). Migrating to using CMMI with its staged and continuous representations will also be a logical choice for those organisations currently using CMM. However, the simple benchmarking levels in the CMM will remain attractive to management in many organisations. ISO9000:2000 will contain added focus on process improvement which may help address the somewhat declining interest in the standard. See [8] for a further discussion of the outlook for SPI.

Case studies – Background

The intent here is not to provide an exhaustive treatment of the experiences of the four organisations. Instead a number of interesting themes of software process improvement will be discussed by using the experiences of one or more of the organisations involved.

NewWorld Commerce (formerly Cunav Technologies):

This is a software systems development and consulting company, which provides IT resources and solutions to customers operating in a variety of application areas, with a focus on web-based development. NewWorld Commerce had approximately 20 staff when they participated in the EU funded SPIRE project [4] that supported focused process improvement projects in small organisations. This case study relates to their experience with SPIRE in 1998 and their experiences since that date.

Motorola Cork

Motorola established the software centre in Cork in 1990 to develop analog switching software and GSM telecommunication systems. There are now well over 400 staff involved in software development. The organisation has had a strong software process improvement programme in place since 1993 (see [5]).

Silicon and Software Systems:

Silicon & Software Systems (S3) have been providing design services in silicon, software and hardware design since it was established in 1986. Within the software division of approximately 100 software engineers (now 150 software engineers), application areas include telecommunications, consumer electronics, Internet and digital broadcasting. The company embarked on an improvement program in 1994 (see [6]).

AIB Bank I.T. Department:

Allied Irish Bank (AIB) has an IT department comprising about 300 personnel providing IT systems and services to the rest of the bank. They embarked on an improvement programme in 1998.

STARTING position and business drivers

NewWorld Commerce performed a review of previous project post-mortems and found some difficulties with managing customer expectations on some projects and excessive amounts of rework due to misunderstanding of initial requirements. Meeting real customer needs and improving project estimates and visibility with the customer were to be the key drivers for the software process improvements. As a small organisation, improvements needed to minimise impact on resources and yet maximise the return on any investments by aligning them with the key business drivers.

Motorola develop systems with high reliability and availability requirements. Given the system requirements and the competitive marketplace, errors and downtime must be kept to a minimum. The telecommunications technology area is constantly evolving, with new products and services continually offered. Time to market is also therefore a key consideration. Motorola's development process is explicitly documented and is evolving as the company follows its program for continuous improvement, which it expects will ultimately lead to an improved CMM rating (Motorola places much emphasis on satisfying the concepts of the CMM).

S3 had a large amount of documentation in their quality management system that required streamlining and a re-evaluation as to the extent that this actually helped people do their jobs. From a business point of view S3 initially stated time to market as the key business driver for the SPI programme. However, it became apparent that the first goal should really be to measure their time to market. This change coincided with an evolving change in mindset re using measurement to guide their improvements. It could also of course not be ignored that achieving a level on the CMM has clear commercial/marketing value for S3 especially since customers were asking for their CMM rating. However, it was made clear that improvement was the goal, not certification to a level on the CMM for its own sake. This distinction proved to be highly significant and beneficial to the entire improvement program.

AIB recognised the importance of managing all changes (including IT changes) as business changes. IT and the business personnel identified a number of key principles/drivers that were to shape the improvement effort:

- IT and the Business were to work together in partnership
- The quality of the product needed to be built-in during development
- Speed to market is a key business driver but it must not compromise the quality of the product and the ability to change into the future

These were combined with specific goals and concerns from the various stakeholders of the improvement programme (including IT staff themselves) to provide direction and focus for the improvement efforts.

The role of the sponsor

The role of the sponsor was of course crucial in each organisation. Without management commitment to the SPI programme the chances of failure are high. Commitment does not simply mean giving approval. It means providing direction, having a good understanding of what is being undertaken and why, providing visible active support and encouragement.

Motorola's management commitment and involvement was demonstrated very well to me on one occasion when we had improved the inspection process with the involvement of a significant number of engineering staff. A key concern raised by the group was whether inspectors would have sufficient time to check adequately for faults given the pressures of project deadlines. The department managers responded by e-mailing all staff that they should notify the managers personally if they found themselves in an inspection where this occurred.

Interestingly, AIB's improvement team initially underestimated the real extent of management commitment that existed. The lesson they learnt was to test the level of commitment early. That way when the commitment is forthcoming it 'kick-starts' the improvement programme with the knowledge that it will be supported by management if/when difficulties arise. (In this case the commitment was very strong. However, if the commitment is not forthcoming it raises the issue early so it can be dealt with as appropriate i.e. halt the programme or work on gaining the commitment!). Similarly maintaining a strong level of management commitment throughout the improvement programme is crucial to its success.

Assessment approach

NewWorld Commerce participated in the EU funded SPIRE project that provided nominal funding for a focused six month process improvement pilot. An SPI mentor was provided to support the organisation during the pilot. The first task of the mentor was to help identify the process area to improve based on business drivers and the results of a facilitated SPICE self-assessment. This facilitated self-assessment involved using an assessment tool to gather data on their processes based on round table discussions with key personnel. It also involved an SPI questionnaire to gather, from a wider group, the attitude towards SPI before and after the improvement pilot. The self-assessment took one day and had minimal impact on resources. It was by no means comprehensive but gave a reasonable indication of strengths and weaknesses for the purpose of the pilot.

Motorola used SEI CMM assessments and its own extension of the assessment approach accredited by the SEI. Between these external assessments, they also performed internal (local) assessments. Interestingly, these local assessments involved comparing the results from the local team's assessment with the results determined by each process area team's assessment of their own process area. This helped ensure a consistent understanding of CMM requirements.

S3 initially achieved ISO9001 certification and then used two approaches to CMM assessment. The initial approach used on two occasions was effectively a facilitated self-assessment. This 'interim maturity evaluation' used a questionnaire and consisted of discussions based on the questionnaire and guided by an external consultant. The training value of these sessions was also described as excellent. An external assessment based on the Process Professional Assessor for CMM assessment method was then performed (this is similar to the SEI's CBA-IPi assessment method)

AIB performed a business review of IT competencies which was a high level assessment of the IT department as a whole rather than a process focused assessment. It then held a number of internal workshops to identify process and organisational issues/problems as perceived by IT staff. Each workshop had a process-related theme or

area for discussion that had been identified as a ‘hot spot’ - this served to direct the discussions. The idea was to identify the ‘pain’ from the bottom up and correlate this with the key business drivers as identified by the business review. Additionally, the CMM was used as a toolbox or reference point to raise questions on the strategy that was planned for improvements. However, this improvement program did not follow a formal assessment approach or improvement model.

improvement strategy

AIB are have been so successful in moving new projects to a Rapid Application Development approach (i.e. using a customised DSDM framework) that they are now using the principles of DSDM on their own improvement project and finding it very useful. These include timeboxing improvement activities (typically one month timeboxes), prioritising SPI requirements and actions within each timebox, using facilitation techniques for SPI workshops, etc.

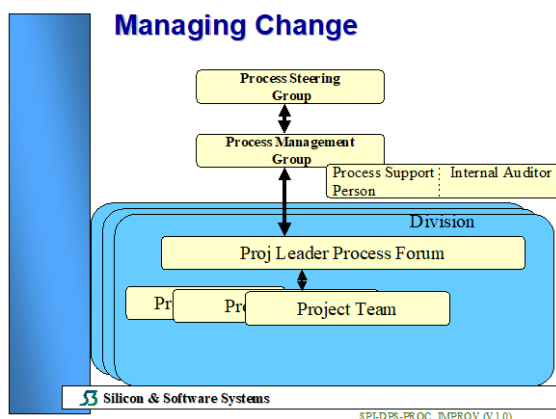
programme management

All four companies followed some form of improvement lifecycle based on a Plan, Do, Check, Act cycle.

Motorola used a variation on the IDEAL² improvement model from the SEI. This is a recommended set of steps to be followed in any improvement programme.

infrastructure

S3 managed to reduce their average time to evaluate and introduce a process change from 533 days prior to the SPI project down to 112 days now using managed pilot projects. A significant factor contributing to this improvement was an infrastructure change (see figure 1.)



² IDEAL is a service mark of Carnegie Mellon University

Figure 1. S3's Software Process Improvement Infrastructure

The Process Management Group consisted of the SPI programme manager, project leaders and key process area team leaders (called task forces in S3 – not depicted in figure 1). The infrastructure change was in response to roll-out difficulties (slow, dissemination of process changes not effective enough, etc.). The key addition was a division-specific project leader process forum. These were key people who are more likely to realise the effect on their projects of any process changes. Improvements that they decided to run with were injected directly into *their* projects where they could assess and monitor the effect of the changes.

training

S3 used a practical approach to training. Initially a oneday overview of the updated quality system was provided which also covered the rationale for changes. This involved a lot of discussion and improvement suggestions were fed back into the system. Process mentors were assigned to each project to support the introduction of new processes. A series of software project management workshops were held that proved highly effective – so much so that it is now done for every new project with a facilitator from the process group. Two project teams participated in each workshop. An introduction to the project was given by the project leader. This was followed by an introduction from the process mentor on the processes that needed to be used/planned (i.e. CMM level 2 processes such as requirements management, project planning itself, configuration management, etc.). Each team then spent the majority of the workshop developing the project plan based on this information. Roles such as project leader were allocated and rotated. This approach introduced people to the quality system, gave the team an understanding of the project leader role and how they could support him/her, was useful as a team building exercise and new projects were doing real work by planning their project!

In terms of external support, S3 used three different consultants during the project that proved useful in providing different perspectives.

Motorola recognise the importance of training and development of a capable workforce. This also helps attract and retain staff in the current climate of high attrition rates. There is a policy of having 40 hours training for each member of staff in Motorola. Motorola made good use of customised training workshops covering a range of topics from CMM overviews to testing and inspections and addressing best practice and the relationship to in-house processes. A key element was the use of in-house projects and documentation for exercises and practicals during training to ensure the training material was related to each participant's own situation.

AIB also used customised training (including AIB's flavour of DSDM, facilitation, Test Management, Peer reviews, etc.) and often held follow-up sessions with the trainer three months after the initial workshop to follow-up on the implementation of the training material thereby enhancing the effectiveness of training.

Cultural/people issues

Winning the hearts and minds of people is crucial to a successful process improvement programme. S3 noted the value of informal communication, especially early in the SPI project, in achieving this – i.e. selling SPI at an individual level. The SPI programme manager coined a key approach used for this... TSDM – the Tea Station Dissemination Mechanism! The key issue, though, in S3's change management and winning people over was having a driving philosophy of achieving real improvement and helping people do their jobs better rather than seeking compliance to a model. Involvement is another key issue – about 70-80% of staff were involved in some manner in the SPI project (e.g. from active involvement in a task force to just acting as a reviewer of proposed changes)

AIB have found that having a strong project focus with direct hands-on support and mentoring of projects by the improvement team has been very effective at enabling effective change. Providing improvement personnel to act as facilitators and mentors on the projects has been the key to this partnership and to viewing process improvement and best practices as a supportive force for the project team. Another key issue has been driving improvements, as least in part, from a bottom up approach whereby improvements are addressing the issues and pain of staff at all levels. At the same time identifying all the stakeholders in the improvement project and ensuring everyone achieves some benefit and has their key needs met greatly increases your chances of success.

One lesson learnt from earlier improvement actions in AIB is the importance of feedback – the loop needs to be closed with people who helped initiate improvements so they see the fruit of their work and have visibility on progress.

Motorola experience is that process ownership and development are best placed with those closest to the process. Empowering staff to define and tailor processes is key to achieving ownership. An example of this was the approach adopted to improving their inspection process [7]. The improvements were defined and agreed in a one-day workshop involving about 35 senior engineers from all departments. This resulted in a great deal of buy-in for the changes.

results obtained

NewWorld Commerce firstly achieved immediate benefits from the pilot projects with the requirements process improvements. This was indicated from data they collected for requirements-related rework and time/budget estimates. However, customers also provided direct feedback on how impressed they were with improved ability to deliver what they wanted. Similarly, the improvements in the project planning and tracking processes (especially in relation to risk management and estimation) resulted in a significant high risk project being delivered successfully.

Motorola achieved level 4 on the CMM scale in 1997. This contributed to the investment in the Cork development group and to its ability to perform highly despite significant and rapid growth.

S3 achieved level 2 on the CMM scale (with a number of level 3 Key Process Areas also satisfied) in 1999. A number of customers (including a CMM level 4 organisation) have performed audits/evaluations on S3 recently and the results have been very positive and have validated the internal improvements undertaken. S3 have even found themselves in the position of being asked by customers to provide advice to them on key

process areas that have been seen to be highly effective! From an internal point of view, a significant change in attitude towards SPI has resulted in S3 staff raising numerous improvement requests – they now see what is possible and realise they have the power to make change and improvement happen.

AIB have performed a number of benchmarking exercises including determining productivity (function points/week), delivery rates (time taken from initial request to delivery to production/function point) and quality (defects/function point). In the words of the SPI programme manager ‘ there has been a huge improvement in each of these three measures’. As significant is the improved way in which people are working together and co-operating.

Lessons learnt

S3 believe that the philosophy they adopted in relation to SPI was crucial to their success. They stressed improvement not CMM compliance. The SPI programme manager is convinced they would not have achieved level 2 (and done so well at level 3) if the goal had been level 2 compliance! Indeed although CMM is a good model, it is just a model and its scope may not be wide enough for a given organisation’s purpose. Another key lesson was the importance of transitioning improvements through a division/department-specific project leader forum to accelerate the rollout of improvements.

Motorola are providing greater emphasis now on using SPI more directly to support the business objectives. In this way even greater gains can be achieved from the use of SPI and leveraging off the progress made in attaining CMM level 4 certification. In terms of improving visibility and tracking progress one must view process improvement as a project – this means applying more structured planning and tracking mechanisms to the SPI project itself in a manner similar to product development projects. Another lesson learnt was the importance of goal-driven metrics and aligning this to business goals.

Conclusions

Key conclusions from these case studies would include:

- stress real improvement not model compliance
- address stakeholders goals/needs...
 - management in terms of alignment to their business objectives/drivers
 - staff in terms of current ‘pain’
- combined process/project workshops may be a very useful just-in-time training approach that integrates process well into projects
- do not underestimate the importance of key roles, such as project leaders, in rapid change deployment
- manage SPI as a project and accessing/utilising SPI experience and skills are critical success factors

ACKNOWLEDGEMENTS

The author would like to acknowledge the kind support of each of the organisations involved in the case studies. The contributions of the following for their help in the preparation of this paper are particularly noted: Bill Culleton from S3, Hugh Ivory from AIB, Tom O’Kane from Motorola and John McEvoy from NewWorld Commerce. Thanks also to a number of people who contributed to the European perspective on SPI: Colin Tully of European Software Process Improvement foundation (ESPI), Tony Elkington of ESPI and Robert Cochran of the Centre for Software Engineering.

REFERENCES

1. Rout, ‘SPICE: A framework for Software Process Assessment, Software Process Improvement and Practice’ August 1995, pp 56-66, ISSN:1077-4866
2. Paulk et. al., ‘Capability Maturity Model for Software, Version 1.1’, Report CMU/SEI-93-TR-24, Software Engineering Institute, Pittsburg, 1991.
3. VASIE Best Practice Repository, The European Comission, DG III Industry, EPSRIT program <<http://www.esi.es/VASIE>> This contains ESSI programme results i.e. information on improvement experiments performed in the European software industry, the results achieved and lessons learnt.
4. SPIRE project and handbook, The European Commission, ESPRIT/ESSI 23873, <http://www.cse.dcu.ie/spire>
5. Fitzgerald and O’Kane, A longitudinal study of Software Process Improvement, IEEE Software, May/June 1999, pp 37-45
6. Kelly and Culleton, Process Improvement for Small Organisations, IEEE Computer, October 1999, pp 41-47
7. O’Hara, O’Kane and Smith, ‘Contradictions within an evolving inspection process’, EuroSTAR conference, Barcelona, 1999
8. Messnarz and Tully, Better Software Practice for Business Benefit – Principles and Experience , IEEE Computer Society (1999), ISBN 0-7695-0049-8

Process Psyops

(Moving a large, well-established company towards process improvement)

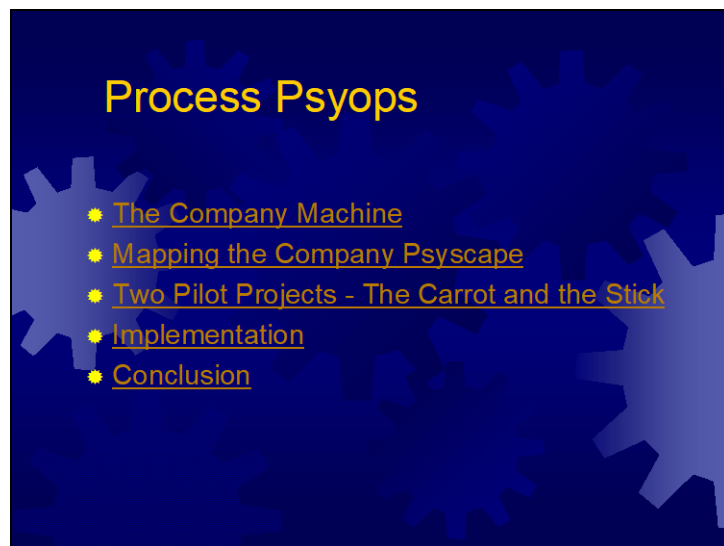
Synopsis

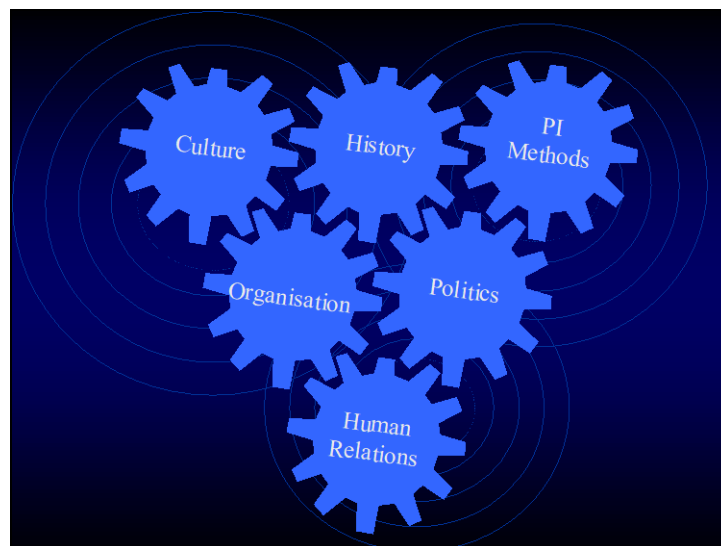
The old adage “It is one thing to know how to read a chart – and another to captain a ship” certainly applies in the case of moving an organisation towards process improvement. Ships range from small, fast-moving and highly manoeuvrable vessels with little cargo capacity to large, cumbersome vessels with enormous capacity. The same can be said of companies.

Moving a large company towards process improvement is a cumbersome affair, requiring a lot of energy in order to build up inertia. Often the problems are not technical or methodical in nature, but are rooted in company culture, history and internal politics. Changes in processes are sometimes so slow that the organisation manages to change two or three times before the process change is accepted, assuming of course that the process change survives the organisational change.

This paper does not provide an easy answer to the inertial problems observable in any large organisation, whether business, military or political. What this paper does is to outline a method which has a higher probability of success than most. Why is this ? Because the method described in this paper allows a small process group, or even an individual to map the psyscape of a large company over a relatively short span of time. The method shows a means whereby “fire support bases” and “pockets of resistance” can be detected and mapped without generating too much political heat. The method shows how two “wedges” in the form of pilot projects can be driven into the company once the mapping operation is underway and provides an answer to the question “why two pilots?”. Finding the weakness of opponents to the process change is a critical operation. Weaknesses vary, but are often centered around a “fire support base” of some kind. It is often politically suicidal to attempt to break the opposition by a direct attack on such bases. There are other ways however. The method describes a means by which an opposition support base can be undermined in such a way that the structure of the resistance to change falls apart next time a frontal attack is faced. The comment “Culture plays a big role” often crops up in large, multi-national companies. The method shows that the role of culture is definitely there, but is secondary to some basic human psychology. Understanding the psychology of development engineers and sales people – two very different types of colleague – is central to process change. This paper shows that despite many indicators to the contrary, it is in fact possible to get the two groups together around a table in a sensible manner, and outlines a simple method whereby the foundations for such meetings can be laid. The question

“What kind of person does it take to be able to use this method?” is answered, and the answer contains pointers to the use of body language, humour, voice and blood-pressure control, apparent aggression and cold-blooded calmness, as well as how to choose and switch between weapons within an instant to obtain a desired effect during a conversation. All humans have defence mechanisms. All defence mechanisms have weaknesses. The paper helps those required to initiate and control process change detect their own mechanisms and strengthen their defences in areas which are weak.





INTRODUCTION:

Large companies are often composed of elements distributed across several countries, and possibly even several sites within each country. Given the international nature of large companies, employees from different countries are often found to be working together in a common environment. Even assuming that each of the individuals involved understand a common language and use it fluently - this in no way guarantees that the company machinery will run smoothly.

In order for the machinery to run smoothly, or even run at all, the cogs must mesh correctly. Any disturbance in this meshing affects the entire machine in

some way or another. Notice the smaller tooth in each of these cogs ? These are disturbing elements which can cause noticeable hickups. Even in a small machine of this kind, with a small number of cogs (development groups) each with a small number of teeth (engineers) with only one disturbing element in each cog, it is apparent that the machine involved will run, but not smoothly.

CULTURE:

Culture is in this case 2 distinct entities coupled under one heading. The first may be described as "National Culture" - summed up in the often heard statement - "They may do it that way on the other side of the Atlantic, but over here we do it differently". The second is "Company Culture" - summed up in the equally often heard statement - "Maybe you did it that way the last place you worked but here we do it differently".

Notice the similarity ? Culture is often used to provide a reason for why things should be done differently in this case. What can complicate the matter is that this is indeed sometimes true. Real differences in national culture must be recognised and taken for what they are - just one more hurdle to be overcome on the way to process improvement. Imagined national cultural differences must also be taken for what they are - imaginary. This does not mean that imaginary national differences are to be ignored - by no means ! But the tools used to overcome imaginary differences are NOT the same as those used to overcome real cultural differences.

HISTORY:

In many ways, history is similar in nature to culture, in that there is a national and a company element involved. In history, however the situation is further complicated by the addition of a third dimension - the backgrounds of the individuals involved in the company, both inside and outside the PEG (Process Engineering Group).

At any given meeting statements will be heard which indicate that there are deep chasms between individuals on some subjects, and common ground between them on other subjects. Statements like "As far as I can see the problem is over here, and we should try this." are often met by the reaction "Maybe, but last time I tried that it did not work." Statements like "Why don't we try...?" may be met by the reaction "Sounds like a reasonable idea - how do we go about it ?". If the participants of the meeting know or believe that the individual making or reacting to the statement has a historical background which supports the statement or reaction, then the statement or reaction may be taken as being reasonable. If not, then other human factors begin immediately to play a more important role.

PI METHODS

The process improvement methods used in a particular company have to be tailored to that company. It is definitely counter productive to attempt to mould a large company into a specific methodology or model just because it has had some success elsewhere. More often than not, each solution involves a complex mix of methods, for example here we need a cupfull from CMM, a teaspoonfull from ISO9000 , a spoonful of GQM. The recipe which has the greatest chance of success depends on all the other factors in this complex machine. For PI people, this means that an understanding of all current and past methods is necessary. It is fatal to allow oneself to be put in a corner from which one cannot escape merely because of adherence to and specialisation in, one particular methodology or model.

In the PI process, it is unfortunate for PEG people to use statements like "According to we must ...". If the individual to which the statement is made is not knowledgeable about PI methods (which is most often the case), then statements like that will leave them feeling diminished, rather than giving them the impression that the PEG person is a knowledgeable individual. When that happens, after the first, second or third statement of that kind within 3 weeks, other human factors start to play a more important role. Such factors may produce counter statements like "That may be so, but what proof do you have that this will help us in this case ?" Once this situation occurs, further attempts to explain PI methods may result in obvious resistance to improvement suggestions. Huge amounts of personal resources will then be required to overcome such methodical resistance.

ORGANISATION:

Because of the complexity, size, production and development inertia of large companies, a typical management reaction to expected or perceived market changes is to reorganise the company. This is most often based on the belief that the organisation of the company is the best means to improve company position in the market, but also on the fact that it is very often easier and faster to give instructions which move people around than to motivate them to new ways of thinking. It is often assumed that once an employee is moved to a new location or position in the company, then that role is filled and the employee will begin to accept the role and play it. As indicated in the section on HISTORY, this is not necessarily the case.

In fact, in production and development environments, too many changes occurring too often, give rise to apathy and resistance to change. PEG efforts to improve processes in these two areas are very often thwarted by organisational changes. It is vital to PEG efforts that the consequences of organisational changes for PI activities are estimated immediately after they occur, and that

steps are taken to limit the adverse effects of these changes on ongoing process improvements. It is also vital NOT to forget that some organisational changes may make some previous process improvements irrelevant. If too many irrelevant processes are prevalent in the company, this also leads to apathy and resistance to change on the part of production and development personnel.

Once the PI machine is running it is possible for PEG efforts to survive organisational changes, but organisational changes occurring during a move of many people to a higher level of process awareness, present a much greater and more catastrophic danger.

POLITICS:

In any company there is a certain amount of political wrangling and manoeuvring between the individuals involved. In large well-established companies this often reaches gargantuan proportions. Politics is special in that most of it happens in corridors and behind closed doors, and involves almost always only management. Although there are indeed politics at engineering and production levels, political discussions here generally have little effect on the company or processes. The effects of political discussions and decisions on PI activities are not always obvious at first, but often become obvious at meetings where PI suggestions are being assessed or results validated. Individuals in PEG's need to be aware of the importance of politics in the crucial role played by management in PI activities.

HUMAN RELATIONS:

Of all the cogs in our machine, this is the most difficult and time consuming to gain control over and indeed to understand. To achieve a high probability of success, it is necessary for PEG people to understand a little of what makes all types of individual in the company tick. Does this take years of experience ? Or does a doctorate in psychology solve the problem ? Can it be done by having been a development engineer, a production engineer and a salesman ?

With human relations governing the foundations of all of the other cogs in our machine, and being too complex to handle for most PEG people, this aspect of PI is often intentionally ignored. Perhaps that is because there is no sure-fire, silver-bullet answer to questions posed and problems raised by human interaction. Even if one has a masters in psychology, sufficient personal experience and an evolved sense of timing, situations still arise in which one must "fly by the seat of your pants" or "play a hunch".

Mapping the Company Psyscape

Large companies normally consist of a number of employees organised into teams with varying functions and sizes.

In order to map a company, it is best broken down into its constituent teams. Each team is evaluated using the method described here. The same method is then used to assemble the team results into an average result for the company. The willingness of the company as a whole to make changes and improve quality is then measurable.

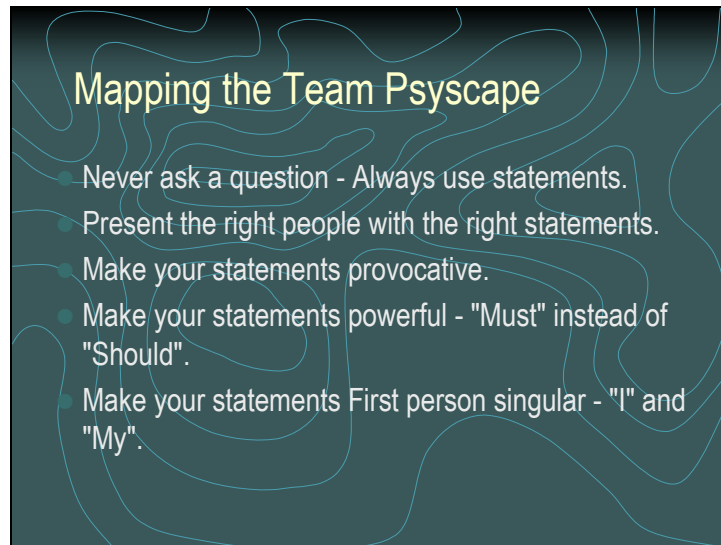
The following slides shown the team evaluation method.

Mapping the Team Psyscape

- Psyscape clarified

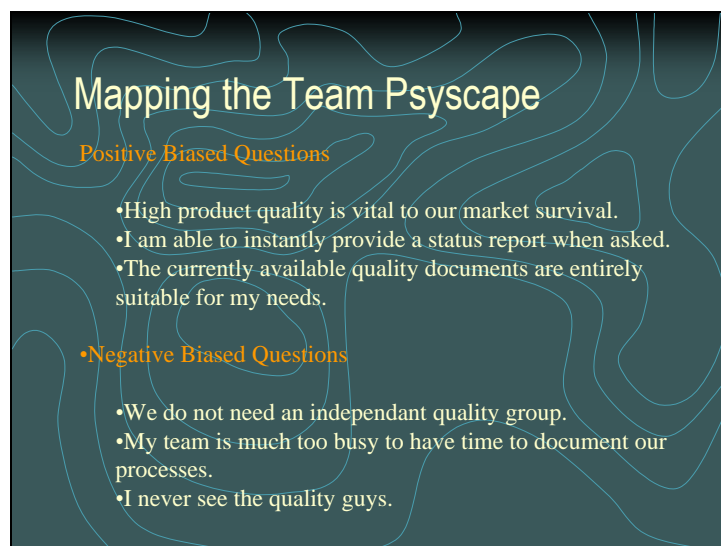
A psyscape is a graphical representation of the opinions of the members of a team.

Once generated, graphics give us the opportunity to detect viewpoints which can pose a hindrance to change, identify areas which need changing and to generate support for such changes.



The mapping operation consists of generating a questionnaire with sufficient questions and a value calculation scheme. The questionnaire is of the multiple choice type with 5 possible responses to each question. In the current model, no weight difference occurs between Middle Management (Team Leader) level and employee level. This may change in the future.

Questions are arranged into two types, positive biased and negative biased. The number of positive contra negative biased questions is random within limits.



This sample questionnaire consists of 77 statements. The possible choices for a reaction are Completely Untrue, Partially Untrue, Don't Care (Irrelevant), Partially True and Completely True. Note that there is no box for "Don't Know". It is all too easy for most people filling out the questionnaire to simply cross such

a box - especially when they are busy and consider the questionnaire as a disturbance rather than as an aid. "Completely Untrue" is given a value of 0 and "Completely True" a value of 4.

A positive biased statement is one in which the ideal answer should be "Completely True". A negative biased statement is one for which the ideal answer should be "Completely Untrue". The ideal value for the reaction to a positive-biased statement is always 4. The ideal value for the negative-biased statement is always 0.

Other than the requirement of sufficient (but not too many) statements, and the sub division of these statements into the two groups, positive and negative, there are no restrictions on the content of the statements. For practical reasons a number of statements covering each Key Process Area of interest should be included, as should be a number of statements covering general opinions.

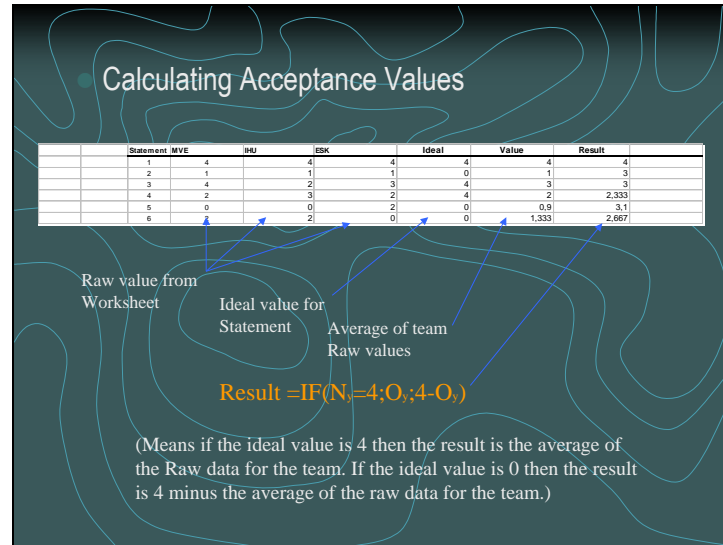
Awarding Values

Number	Statement	CU	PU	DC	PT	CT	Ideal Value	KPA
1	My product quality is vital for market survival						4	Requirements Management
2	I know precisely what my customers are						4	Requirements Management
3	I know exactly what my customers want						4	Requirements Management
4	My customers and I always misunderstand each other						0	Requirements Management
5	My customers don't know what they want						0	Requirements Management
6	We almost always fulfill our customers' functional needs						4	Requirements Management

CU = Completely Untrue
 PU = Partially Untrue
 DC = Don't Care (Irrelevant)
 PT = Partially True
 CT = Completely True

An ideal value of either 0 or 4 is attached to each statement. This value is not visible to the subject filling out the questionnaire. Whether the value is 0 or 4 depends on whether or not the statement is to be taken in a negative or a positive context.

The sample questionnaire is in the form of a spreadsheet containing automation macros and suitably protected so that the subject only sees the statement and their possible answers. The subject crosses one box for each statement and returns the questionnaire. In this sample case the questionnaire takes about 15 minutes to complete.

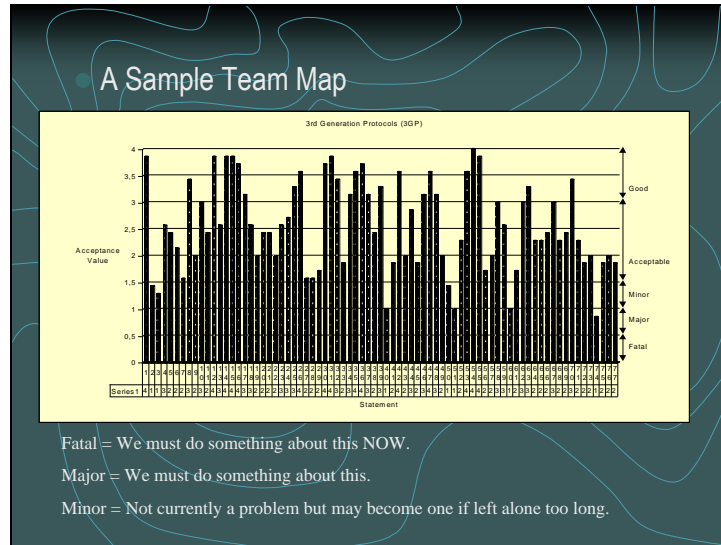


The questionnaires returned by each member of a team are saved in a separate spreadsheet as reference material. Macros transfer the answers from the subject spreadsheet to the team result spreadsheet.

Further clarification of deviations requires manual intervention. Clarification is required because on some occasions, a result is deviant because the person reading the question did not completely understand it. If this is the case then the question may need rephrasing in order to increase user understanding. On other occasions the deviation may be a genuine disagreement with the statement. In these circumstances the result is to be taken as genuine. It is quite legal for a subject to change the value of an answer after a round of clarification, but not legal for a subject to change the value of an answer if the question was understood originally and the answer genuine at the time the questionnaire was completed.

As results arrive, the graph representing these results will change. A minimum of 60% of team members results will be required to make the data reasonably reliable. Groups with under 5 members should be avoided, but since most groups in a large company are equal or larger than this, then the group size in these cases is not a practical limitation. If more than one group of under 5 members exists, these can be combined for the purpose of the questionnaire, but even more care must be taken in the choice of statement and groups have radically different functions should preferably not be combined.

Ideal values must only be 0 or 4 so statements must be arranged so that only one of these values is ideal. Ideal values of 1,2 or 3 are illegal.

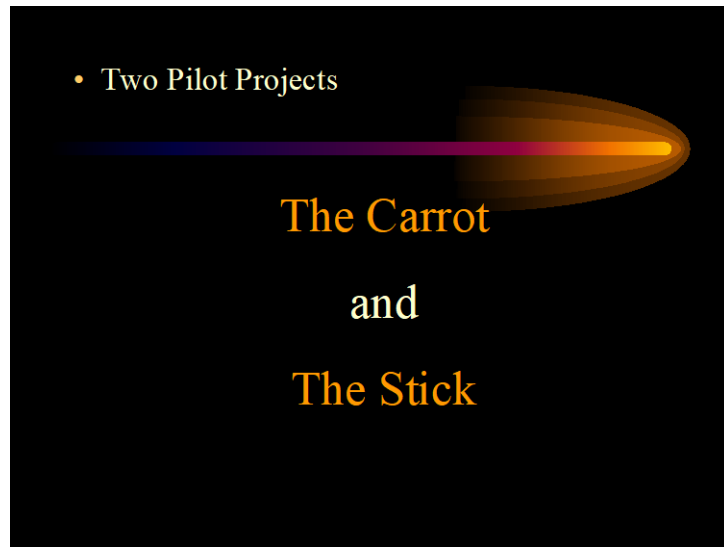


The ACC values from the team for each question are placed in the reserved column on the graph.

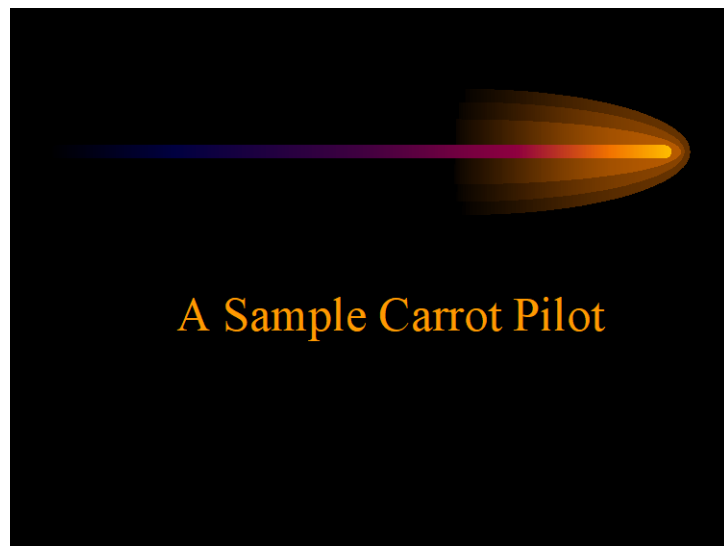
The maximum value achievable in each question is 4. The minimum is 0. The graph is divided into 5 regions. Results below 0.5 indicate statements with which the team completely disagrees. Assuming that the statement is understood this represents a "Fatal" problem. Values between 0.5 and 1.0 indicate a "Major" problem. Values between 1.0 and 1.5 indicate a "Minor" problem. Values between 1.5 and 3.0 indicate an "Acceptable" condition, i.e. no basic disagreements. Values between 3.0 and 4.0 are "Good", meaning the team accepts the statement fully.

When the results are available, take a meeting with the team leader and discuss them before finalising your comments. This is necessary to ensure that the results have been interpreted correctly.

For example, an entire team may answer "Don't Care" to a specific statement and score a low level. If the statement is in fact irrelevant for the team then this is acceptable. If however, the statement is relevant for the team and the team members do not care, then the team has a fatal problem on the horizon which should be dealt with as soon as possible.



In any company of appreciable size two very opposite types of team will always be represented. The first is the team which is highly motivated for change, the second is the team which is not motivated for change at all. It is human to try to avoid problems - to play down the role of those who are against change and to concentrate on those who accept change suggestions more easily. However, in order to make change possible we **MUST** take the second type of team into account as well as the first type. In other words we must not ignore the impact of resistance on what we are trying to achieve.



The Carrot Pilot

The first pilot project is selected and defined. This project is centred around subjects and personnel which, through the answers given in the questionnaire,

indicate that the probability of successful change is high. Teams of this kind provide a beacon in the change scenario, pointing out to others less motivated that it is indeed possible to reach the target, and are therefore referred to as "FSB" (Fire Support Base).

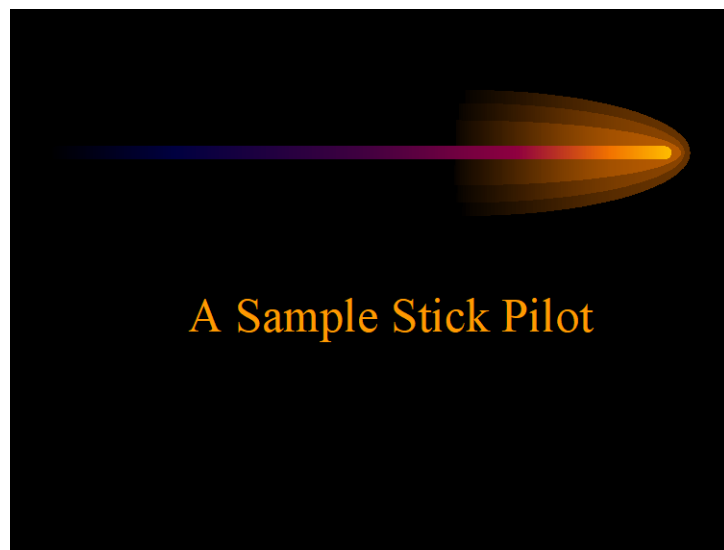
- Select the already established team which shows the highest acceptance figures for all the involved individuals as the Carrot Pilot.
- 2. Assemble the whole team at a meeting, and describe to them the terms and conditions of the pilot project, in addition to its targets and expected time frames. Have the team select an individual who is to be responsible for QA activities within the team.
- 3. Discuss the processes currently in use in the team. The QA person is then required to make a detailed presentation of these processes, have the presentation reviewed and accepted by the other team members. It is vital that the finished product reflects the world as viewed inside the team and does not contain ideas for changes to their current processes.

In this example, it is assumed that the organisation in question wishes to raise the level of its R&D department from CMM level 2 to CMM level 3. This is perhaps the most difficult step to achieve because of the psychological gap between level 2 (Repeatable) and level 3 (Described) and the resources required to make the move.

- 4. The team leader, the QA person and the PEG member discuss the results. During the discussion, the process is checked for conformance to CMM2 and to CMM3. Any missing points or required activities are merged into an Action Plan.
- 5. When all items on the Action Plan are completed, the whole team is assembled and informed of changes in the processes which are the result of performance of the Action Plan. Questions regarding changes are answered and obligatory use of the new process description is initiated.
- 6. A simple self-assessment is performed to indicate that CMM3 is attained. The group process is raised to a standard. From this time on no process changes may occur without consent from the PEG.

Pilot Project Completion:

The carrot pilot phase may be considered as being complete when a new questionnaire filled in by the participants in the shows that the acceptance level has at least remained unchanged. During the pilot phase it is likely that the carrot project reaches its target first before the stick pilot, but this is of less importance than the depletion of resistance to change within the organisation - the purpose of the stick pilot. Do not use the same questions in the second questionnaire as were used in the first questionnaire.



The Stick Pilot

The stick pilot project consists of a grouping of individuals, all with the same or similar viewpoints which supports their resistance to, or lack of sufficient motivation for change. Viewpoints of this kind are referred to as a "POR" (Point of Resistance).

It is not the purpose of the Stick Pilot project to gain obvious success by "converting" disbelievers to believers. The purpose of the stick pilot project is to undermine the resistance to change represented by the viewpoints coagulated into the POR, by forcing the participants in the project to discuss and consider some of the points in the POR, and to document their recommendations. The stick pilot requires much more attention and energy from the PEG, since the recommendations will often be lacking or negative.

1. Select the already established team which shows the lowest acceptance figures for all the involved individuals as the Stick Pilot.
2. Assemble the whole team at a meeting, and describe to them the terms and conditions of the pilot project, in addition to its targets and expected time frames. Have the team select an individual who is to be temporarily responsible for QA activities within the team.
3. Discuss the processes currently in use in the team. The QA person is required to make a presentation of advantages and disadvantages of these processes, have the presentation reviewed and accepted by the other team members.

4. The team discusses the disadvantages, possible solutions and which effects these solutions have on what was previously regarded as advantages of their processes.
5. The team discusses which effect CMM should have on their work. During this process it may happen that another QA candidate with a higher acceptance level may appear. The PEG representative should then replace the previous QA person with this person.
6. Continue with 4 and 5 until it is felt that resistance has decreased sufficiently to warrant filling a new questionnaire. Then have the team do so.

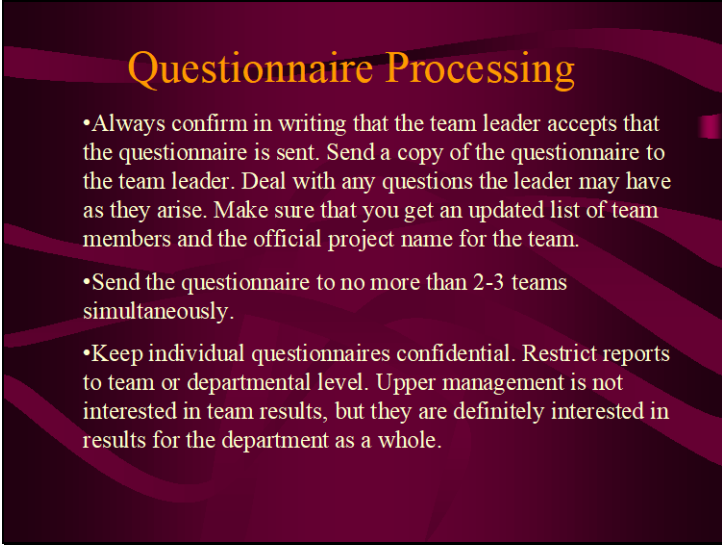
Pilot Project Completion:

The stick pilot phase may be considered as being complete when a new questionnaire filled in by the participants in the pilot shows that the resistance level has decreased to the vicinity of indifference (acceptance level 1.5 on the psyscape). Do not use the same questions as were used in the first questionnaire.

Implementation

- Questionnaire Processing
- Public Project Introduction
- Follow through to completion
- Tying up loose ends

Questionnaire Processing

A presentation slide with a dark purple background and wavy patterns. The title 'Questionnaire Processing' is in a gold, serif font. Below the title are three bullet points in white text.

Questionnaire Processing

- Always confirm in writing that the team leader accepts that the questionnaire is sent. Send a copy of the questionnaire to the team leader. Deal with any questions the leader may have as they arise. Make sure that you get an updated list of team members and the official project name for the team.
- Send the questionnaire to no more than 2-3 teams simultaneously.
- Keep individual questionnaires confidential. Restrict reports to team or departmental level. Upper management is not interested in team results, but they are definitely interested in results for the department as a whole.

It is important at this stage that the leader of each team feels that he/she is not losing control of team resources. Remember that the team has other things to do than answer questionnaires, and if any of them are in doubt they will contact their leader. If the leader is not in full agreement with using resources, no matter how small, then the project will stop there.

A number of questions and comments can be expected when the questionnaire is sent to teams in the field. Hopefully most of the bugs will be removed from the questionnaire before it is sent, but there will always be a small number to deal with. Sending the questionnaire to no more than 2 teams simultaneously allows a reasonable rate of progress while at the same time allowing time to deal with any situations which may arise.

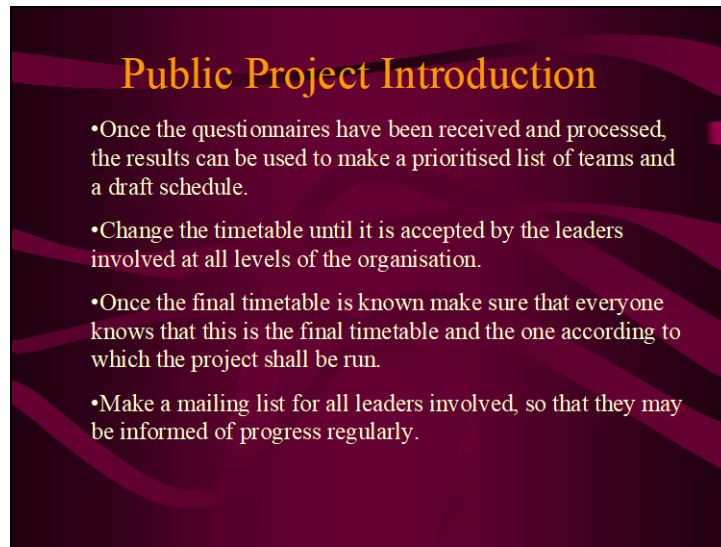
Team results should be returned to the team as soon as possible after the last reply has been received. Give the leader a few days to assess the results then take a chat with the leader to find out if he/she has any suggestions for improvements if these are needed. Spreadsheet automation makes it possible to provide departmental status reports within minutes of receiving a request from upper management.

Questionnaire Processing

- Present the results to the team leader immediately they are available. Do not comment the results as comments may have the opposite effect to that which you desire. Simply state which statements are in Fatal, Major and Minor condition. If there is anything to say about the results then say it personally to the team leader, not in the report, which is public.
- Most required functions for report generation can be automated using spreadsheet macros. An automated spreadsheet has been developed in connection with this project, and can be made available on request.

It is vital that the team leader and the team feel that things are happening at the right pace. If there is too much pressure on this subject in relation to their specific work, they may develop a negative attitude to the entire process because of the apparent overload on them. If nothing happens for several weeks they may also develop a negative relationship with the project because they feel that it is a complete waste of time. Strike a balance. If something happens with each team once every other week, even if it is only that you turn up to have a chat about how things are going, they will feel that they have your attention. Make sure that you say time and time again that your door is open and they can come for a chat when they need it. When one eventually does - be there!!

Public Project Introduction



Choose the teams with greatest probability of success (greatest Team Index - the number calculated from the results of all team members) as the top priority. Prioritising according to the Team Index makes the choice of teams simpler.

By this time the vast majority, if not all, of the people who will be involved in the change will have had a questionnaire and some form of contact, either personal or otherwise, with the group responsible for the change. Certainly all the team leaders will have had several contacts with the PEG. So an official announcement of the plan should not generate a great deal of surprise or conflict.

The announcement then need not take a high probability of conflict into account, but should concentrate more on describing the process to be followed, the aims of the process, the expected resources to be required by the change process and the advantages/disadvantages of the final result. It is also necessary to provide some kind of draft timetable on which the time frame for each team is indicated. Do not leave anyone out, no matter how minor their role seems to be ! Also, be prepared to change the timetable until there is general agreement among the leaders involved that it is the most acceptable result. This may take some time, but do not be tempted to get on until this is in place. Do not attempt to move more than 2 groups at a time per person in the PEG, since there will be a lot of discussion involved and much personal attention required.

Public Project Introduction

- Announce the change project to all levels of the organisation. It is advantageous at this stage to have a very high level "mentor" to promote the work of the group to the remainder of the organisation at the time of the announcement.
- Allow sufficient time for comments and answers from the organisation before the start of the project, and expect such comments and questions. Deal with them quickly and make sure that the originator of the comment or question is satisfied with treatment received, even if they are not satisfied with the answers you have to give.

A logical choice of mentor is the highest person responsible in the organisation. In very large and divisionalised organisations, this may not be required or possible. Choose the highest possible and relevant person in the part of the organisation making the change. Present your plan to this person carefully, even though it is highly likely that this person has been following the development of the plan from the very beginning, and may even have initialised the entire operation. Remember that you will be providing the ammunition this person requires to publicise the change in a language and manner which the rest of the organisation will be able to accept.

Follow through to Completion

Follow through to Completion

- The method to be used during implementation is that stated in the carrot pilot project description. Once the decision has been made to move the organisation to a CMM level, the stick pilot project, which has of course been completed prior to the top level decision to move, has outlived its usefulness. The lessons learned from the stick pilot project will of course be very useful indeed to the PEG during the practical implementation of the plan on a daily basis.

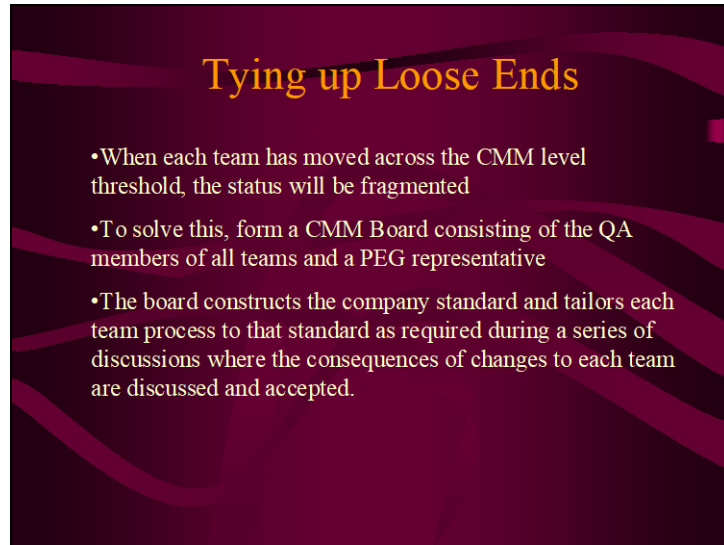
Follow through to Completion

- In most organisations, tracking effort usually means reserving an account number of some kind specifically for this operation. Ask the team leaders of teams being moved to use this account number for all activities connected with the change.
- Keep track of your time and resource estimates using the account. Each time a team passes the required CMM level threshold, publish your measurements of time and resources for that team and how they correlate to your estimates. This may mean amendments to estimates and plans for other teams so be specific and amend the master schedule accordingly.

Follow through to Completion

- Before the last team crosses the CMM level threshold, it is time to begin specific planning of an external assessment. Choose the most relevant assessing organisation for your company and discuss the matter with them.
- Once the external assessment is completed, it may become obvious that the organisation is still lacking in certain process areas and that further work is required. This is another situation, outside the scope of this paper, but will most likely consist of following up on the action points agreed with the external assessors, and calling them back for a new assessment when sufficient action has been taken.

Tying up Loose Ends



When the teams have apparently all crossed the CMM threshold, the organisation will still not be ready for CMM assessment. This is because the status of the organisation as a whole will be fragmented, each team having its own processes and there being no organisational structure common to all teams.

During the work done to move to the CMM level, each team selects a person to be responsible for QA activities within the team. These persons can be collected together along with a PEG representative to correlate the team processes, and to construct a standard company process from those which are by now documented. Although this is in the opposite sequence to that recommended by CMM - where a sub set of processes will be a tailored version of the company processes, this condition is still fulfilled, because the company processes are a standardised form of the prevalent team processes, which again are tailored to match the company standard. The Board will decide which amendments need to be made to the processes of each team to make the whole arrangement as standard as possible, and being the QA responsible for a team, puts each board member in the correct position to be able to initiate process changes within the team when agreements have been reached.

Conclusion

The published method will enable a PEG to move a company with no previous CMM experience, or a company with a lower CMM level with the will to move to a higher level, to do so. The estimated project time will of course be dependant upon the number of teams and the acceptance level of each team and organisation management, but for somewhere in the region of 100 developers a project time of 24 months will not be unrealistic.

The question remains "WHAT THEN ?".

Whether an organisation moves to an even higher CMM level, or diverges into CMMI depends heavily on the strategy and customers of the company.

The situation is a new one and is outside the realms of this paper, but once the published method has been institutionalised, a tailored version of it can be used to take whichever path the company upper management sees fit to take.

Whatever else happens, in development companies, the PEG machine cannot afford to stop and rest on its laurels.

Session 3 - SPI and Testing

**Session Chair:
Miklos Biro, Sztaki, Hungary**

Quality Assurance for Software in Context of Supplier's Management	3-2
Test Execution and Test Management for Numerical Control Software	3-17

Quality Assurance for Software in Context of Supplier's Management

Uwe Hehn

method park Software AG i.G., Erlangen/Germany

Bernd Hindel

method park Software AG i.G., Erlangen/Germany

Abstract

If a customer obliges its contractors to use software delivered from his software suppliers, he has to make sure the necessary quality of this software – in the following called "passed-on" software. Otherwise problems as strongly growing costs and significant delays will result. So it is the customer's own interest to be able to assure a high quality of the "passed-on" software. However this may be a difficult job, particularly if the customer does not utilize the software himself, and does not regard the implementation and verification of software as a domain of his core competence.

In the first part of the paper the basic possibilities of quality assurance a company may deploy in the scenario are outlined. Then a suitable combination of these possibilities is proposed, which allows an efficient quality assurance for "passed-on software" with reasonable effort.

Introduction

The following scenario shall explain and motivate a situation often occurring in many areas such as, for example, the automotive industry or industrial automation this situation (Fig. 1):

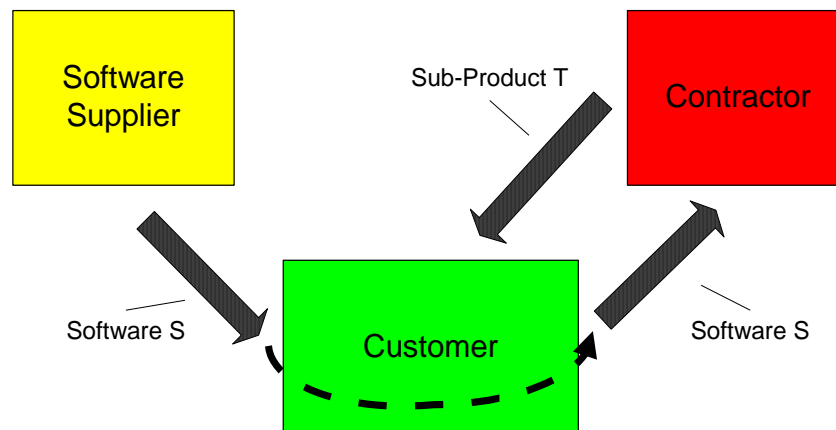


Figure UHEHN.1: Customer passes on software bought from a supplier to a contractor

- A company (the *customer*) commissions its *contractor* to build and deliver a sub-product T, which the customer will then integrate into its product.
- The customer obliges the contractor to use the software product S – made by a third company, the *software supplier* – for the production of sub-product T. For this purpose, he provides the contractor with the software he received from the software supplier.
- Before *passing on* software S to the contractor, the customer wants to make sure that this software is as error-free as possible, complies with the given interfaces, etc., in order to avoid
 - a) complications arising on the contractor's from the use of software S during the production of sub-product T and
 - b) serious problems and delays during the integration of sub-product T into the product.

As the distinguishing property of the software regarded here is, that after delivery by the software supplier it is passed-on by the customer to the contractor, this kind of software shall be called "passed-on software" in the following.

Oftentimes, not only the software of one software supplier is used but also several software suppliers will provide software packages S1, S2, ... for the production of sub-products (see [3]). On the other hand, several sub-products T1, T2, ... will be built by several contractors using software packages, S1, S2, ... (Fig. 2).

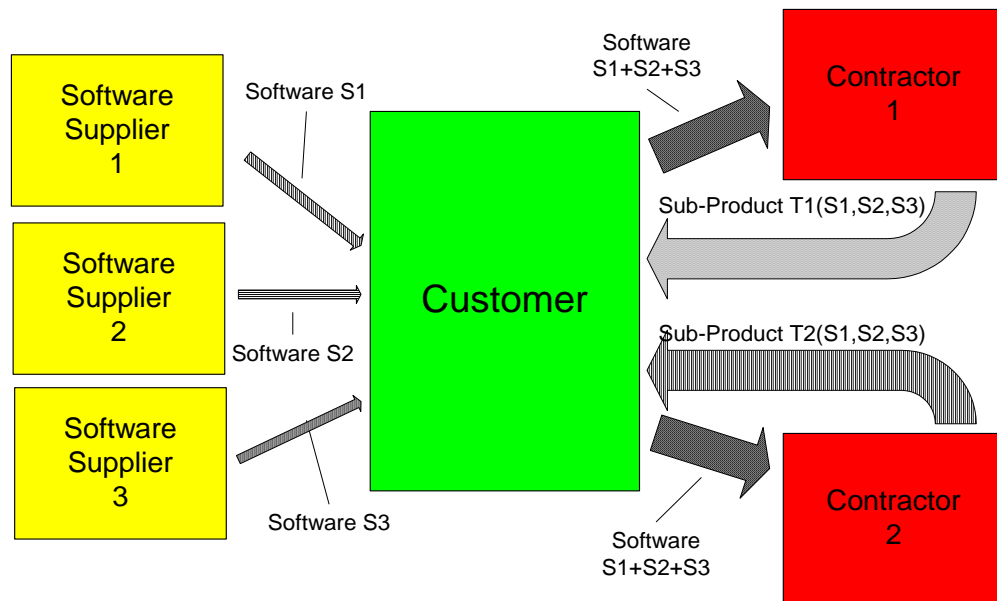


Figure UHEHN.2: Contractors use software provided by several software suppliers

The customer has to solve this task successfully "with adequate effort". This means especially that

- not all basically feasible measures can be applied,
- redundant test measures must be avoided.

The solution will be for the customer to use a clever combination of possible measure to achieve good coverage with a given "adequate effort".

Basic Intervention Options

We first look at the simplified situation where the customer receives software S from only one supplier and passes it on to contractor Z for use during the production of sub-product T (Fig. 3).

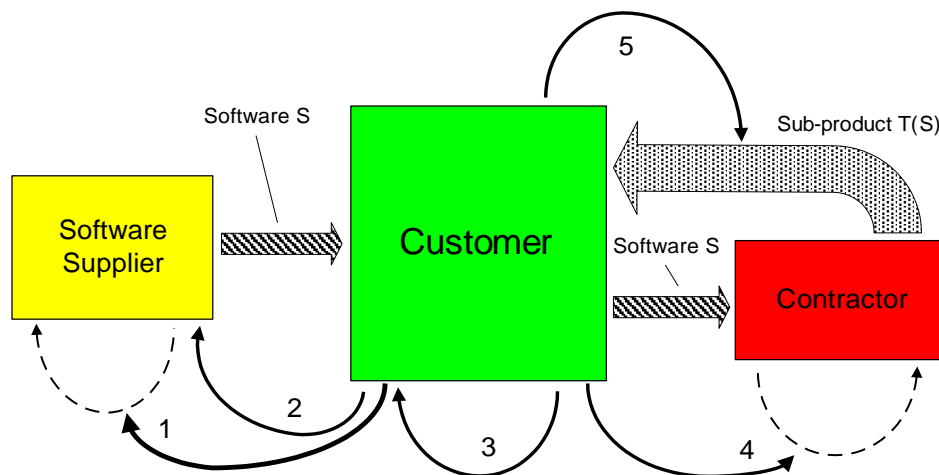


Figure UHEHN.3: Basic intervention options of quality assurance

The customer can choose between the following intervention options:

- 1) The customer trusts the software supplier's own quality assurance.
- 2) The customer influences the software supplier's quality assurance.
- 3) The customer tests the software received from the software supplier before passing it on to the contractor.
- 4) The customer judges the quality of the software using feedback from the contractor.
- 5) The customer indirectly tests the quality of the passed-on software during the integration of the sub-products into the product.

Remark:

Furthermore the customer could perform a risk assessment for each of the external software components ("How can they comprise the system by failing in a certain way?") and then write wrappers that prevent these things from happening. This idea is used in safety critical software. Trying to deploy this method with regard to "passed-on software" would be interesting, but this would go beyond the scope of this article.

The following examines the usefulness of these options (1-5).

Customer Uses the Software Supplier's Own Quality Assurance

The customer assumes that the software supplier has defined an adequate software development process. Parts of this well-defined process are a precise requirements description as well as design and interface descriptions.

A testing concept based on the requirements, correctly filled-in testing protocols and review protocols of the document and code reviews conducted during the tests document the quality of the software supplier's software development process. Proof of this can be supplied with a mandatory confirmation given by the software supplier.

The software supplier can also have the tests conducted by a different company offering testing services.

Moreover, the availability of a validation suite for the judgement of the delivered software comes in very handy. A successful validation by an organisation such as the German TÜV ("society for supervision of technical systems") is another confidence-building measure.

If the delivered software is an established and proven standard product available in a same or similar form, and if the software has proven its functionality and reliability in wide-spread use, this kind of quality control is usually considered adequate. Typical examples are compilers or operating systems that have proven their quality for certain platforms.

Customer Makes Quality Demands on the Supplier

The customer has to define quality requirements for the custom-made software or the software specially configured for the current project. Furthermore, the customer will expect a good software quality from a certified software process.

An accepted certificate (e.g. CMM level [1], SPICE [1]), can be required as proof for the maturity of the software development process; an effective testing process can also be verified by a successful audit corresponding to a test-process-improvement model (e.g. TPI [2]).

The quality control measures undertaken by the software supplier – e.g. in the context described above – should be revealed to the customer; this is especially true for all testing methods and results. If desired, an available validation suite and all documentation pertaining to the software development process must be provided to the customer.

The Customer Tests the Software Provided by the Software Supplier

Theoretically, it should be possible for the customer to directly assure himself of the quality of the delivered software. Oftentimes, however, the customer does not have the facilities for conducting his own tests or other inspections, because, e.g. he does not have the necessary software development or testing know-how and he does not plan to engage in building up. Only if software development is one of the customer's main competences, he will really be interested in performing the test by himself.

The following approaches are theoretically feasible:

- Conducting tests with support from the software supplier
- Conducting tests independent from the software supplier
 - a) in the customer's own test lab
 - b) contracting the tests to an external test lab

Measures in order to evaluate software quality

In general the customer can apply the following measures:

- static inspections (static code analysis, reviews)
- dynamic tests (e.g. running an available validation suite, provided by the supplier or a neutral organization)

In general, dynamic tests can only be conducted in special environments (e.g. the software suppliers' development or testing environment). This is especially true when the software also accesses project-specific hardware. Dynamic testing outside of the software supplier's development/testing environment is very costly. Therefore, if the customer insists on conducting his own dynamic test, this testing should be done in cooperation with the software supplier, and within his environment, if possible.

Static testing does not require special technical efforts. In principle, static tests can be conducted without the assistance of computers, even though the use of static analysis tools can decisively increase the efficiency of static tests. Since computer-assisted static analysis is independent from the target platform, it can be conducted on a suitable available standard platform.

The customer may subcontract the execution of both dynamic and static tests to a software/test house that is independent from the software supplier.

Customer's Quality Control Using Feedback from the Contractor

If the contractor finds deviations in the software he is to use, these deviations must be analysed carefully: Is there really an error in the delivered software? Has the software been used incorrectly? Sometimes, this distinction is difficult to achieve and can require a lot of time. If an error is indeed found in the software, the delivery of an improved version will take some time, which can lead to a significant delay in the planned deadlines.

Of course, it is preferable for the contractor to find any remaining errors, rather than accepting those errors as a ticking "time-bomb" in the sub-product. Nevertheless, it should be attempted to find any possible errors before the software is passed on to the contractor – among other things to avoid additional claims made by the contractor. As is generally known, fixing errors gets more expensive the longer the time between its formation and its exposure and removal is. Costs as well as the required time especially, increase exponentially, which can have serious consequences.

Besides a costs explosion and drastic delays, the customer's good reputation might also suffer, which makes an effective quality assurance even more important.

Customer Indirectly Tests the Delivered Software during Integration of Sub-Products

This situation widely corresponds to the one described in "Customer's Quality Control Using Feedback from the Contractor" above. Since possible errors are found even later, however, the above-mentioned problems will get even worse (costs explosions, delays, loss of reputation).

More Than One Software Supplier

If there are more than one software suppliers, the situation is similar to that with one supplier with the additional problem that the software packages S1, S2, ... delivered by the different suppliers generally must cooperate.

By adding a software integrator, the situation "more than one software supplier" can be reduced to the situation "one software supplier".

Efficient Intervention Options for the Customer

After the general intervention options available to the customer have been clarified, the following describes a sensible concept for a quality assurance of the "passed-on software" from the customer's point of view. (The numbers in round brackets used in this section refer to figure 3 respective figure 4.)

Influence Too Small

The alternative "software developer's own quality control" (1) only applies to standard products where the customer can exert only little influence. If the customer needs more control, he should negotiate with the software supplier on that account. The software supplier, for example, can give the customer access to his quality assurance methods, or agree upon a joint validation of his standardized software. This, however, changes the customer/software supplier relationship towards the alternative "customer makes quality demands on the software supplier" (2).

Influence Too Late

The influence options given by "feedback from the contractor" (4) and the "indirect test during integration of the sub-products" (5) do not help to prevent errors and problems. Errors are discovered at such a late point in time that significant consequences regarding costs and time schedule must be expected: the removal of errors in late phases generally is costly; the probability that the software contains even more undiscovered errors is high.

Therefore, feedback from the contractor and quality assurance during the integration of the product are not really suited to achieve a good quality standard for "Passed-On-Software".

Realistic Influence Options for the Customer

Based on the above conclusions, only the following influence options remain to the customer for an efficient quality assurance (Fig. 4):

- The customer actively influences the software supplier's quality control (2).
- The customer subcontracts the integration of the individual software packages and the validation of the successful integration to a software integrator (2a)
- The customer himself tests the software received from the software supplier before passing it on to the contractor, or subcontracts this task to another company (test lab) (3).

These control options are combined in a suitable way to achieve a good quality assurance.

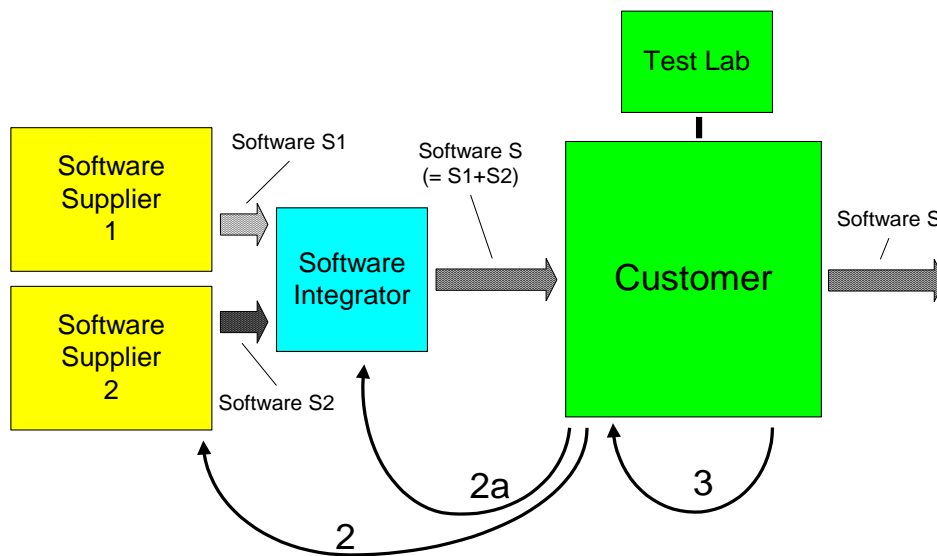


Figure UHEHN.4: Workable possibilities to assure the quality of "passed-on software"

Putting the Concept into Practice

The following assumes that even though the customer wants to achieve a high quality of the "passed-on software",

- the customer can afford only limited expenditures
- the customer does not want to or is not able to conduct his own tests, if they are not available in the form of a very-easy-to-use test suite
- and that the relative costs for implementing the suggested process decreases as such tests are repeated; i.e. a one-time expenditure for establishing the testing method is acceptable, the tests of regular updates or new versions, however, may require only little time and material costs.

In the following the subject "dynamic tests" is kept only very short. The focus is put on the "review" or "static" approach.

Dynamic Tests

Dynamic tests are used to reveal discrepancies in the execution of specified test cases. Since the tested software is to be used by different contractors for the development of diverse application software, i.e. the software will be used in various different ways, this must be considered during the software test. Drawing up several typical use scenarios that are made available as a test suite can do this. As soon as the test suite is available, the tests may be performed by an independent test lab or by the customer himself.

Requirements on such test suites are

- ease to use by a user without special test know-how ("select test case, start test, get the result")
- and maintainability and extendibility by experts ("new test cases, changed test cases")

Because of the complexity and variety of the systems under test (for example in the automotive industry many different operating systems, many different processors and many different boards have to be considered), suitable test suites are not generally available but have to be developed tailored to the needs of the customer. Test suites of this kind are very powerful but typically very expensive, too.

(Some such customer-specific test suites have been developed by 3SOFT recently and successfully used by the customers.)

Code Reviews

The following summarizes the different versions of code tests such as desk test, code inspection or code review (see [4]) under the label "code reviews".

Code reviews are a very valuable tool for testing and evaluating the source code of software, Together with dynamic tests, which can reveal selective deviations during the execution of specified test cases, code reviews can be used to check functional correlations and the technical quality of the developed software.

Depending on the participants in a code review, different emphases can be observed. We take a closer look at

- code reviews being conducted at the software supplier's, and
- code reviews being conducted at the customer's or in a test lab authorized by the customer.

Code Reviews Conducted by the Software Supplier

The main focus of code reviews conducted by the software supplier is determined by the following parameters:

- Code reviews are being conducted from the point of view of software development (the software developers participate).
- Code reviews can be conducted both during the development phase and at the completion of a software unit's development.

- Development documents, such as system design and module design, are available to the participants of the code review.
- The low-level aspects of the implementation can be tested competently.
- It is possible to run through use scenarios (testing of the functionality).

The customer should demand proof that the code reviews were conducted (review protocols). If necessary, some of the customer's employees can participate in the code review as observers.

Code Reviews Conducted by the Customer

In contrast to code reviews conducted by the software supplier, code reviews conducted by the customer are determined by different criteria:

- Normally, the software developers who worked on the software will not participate (this makes it impossible or difficult to ask them questions).
- The test will encompass the complete software released by the supplier.
- Testing the functionality is not possible or requires great effort since the required developer's know-how usually is not available to the customer.

The customer can subcontract the code reviews to a test lab but in general, the given criteria also hold for this situation.

Static Checks

Normally, the customer will accept the test protocols and affirmations provided by the software developer if they guarantee both the required functionality and the non-functional characteristics.

If necessary – e.g. when the suppliers increasingly have problems – the customer may, however, – based on respective agreements – inspect the software supplier's software development process as closely as desired. This makes it possible for the supplier to basically get by without his own complete test of the delivered software. Of course, the transfer of the concept described here can be supplemented with other analytic methods – such as automated regression tests –, which makes them even more effective.

Nevertheless, the customer should not dispense with testing the outer appearance of the delivered software – using static analysis – since this makes it possible to draw substantial conclusions about the care taken during the development of the software.

The following analogy is to serve as an example:

For somebody who is not a technical expert, it surely is difficult to adequately evaluate the "insides" – the components and aggregates – of a modern automobile. Nevertheless, it is rather easy to receive an appraisal of the vehicle from the user's point of view:

- *A test drive (corresponding to using an "acceptance test suite" gives an important impression of the automobile's functionality.*

- *A look under the hood or the chassis ("static tests") quickly reveals the care taken to build the car: "clear construction and clean screw joints" or "tangle of cables and wires and shaky parts"?*

The following describes the static tests that from the customer's point of view make sense for testing the "passed-on software". An important principle is the automation and easy evaluation of these tests.

Static Checks with Tool Support

The following are examples for relevant checks that can be conducted statically with a relatively low effort:

- observance of development guidelines
- adherence to naming conventions
- testing whether documents and source code consistently contain requirement keys
- testing for adherence to the interface description.

This list can be continued as needed.

Commercial tools are available for these kinds of tests. The capabilities of the respective tool and the possibilities available for parametering and adapting the tool determine which kind of results can be expected. .

Evaluating Metrics

Metrics tools implement a special kind of static analysis. Software metrics numerically record characteristics of the inspected software. A multitude of commercial tools is available for calculating metrics. In order to be able to use metrics for the quality assurance of "passed-on software", sensible, acceptable value ranges must be defined for each one of these metrics.

Complexity

Among others, the following metrics are used to evaluate complexity – the list can be continued as needed:

- number of branches
- nesting depth
- number of classes/methods
- size of functions
- size of files

Modularisation

The following metrics may be used to evaluate the degree of modularisation:

- How many global variables are used per source code file, per module, etc.?
- Are global variables only accessed via access functions?
- How large are module coupling (measure for the use of module-external functions and variables) and module cohesion (measure for the use of module-local functions and variables within the module)?

”Outward Impression”

The ”outward impression” of software can be evaluated with metrics like

- share of comments
- source code density.

Static analysis can also determine whether the control structures are indented correctly. Of course, a manual code review is necessary to test such criteria as ”the comments make sense”.

Prerequisites for Effective Code Reviews

What can the customer do to effectively use the results of the static analysis or the determination of metrics? (s. Fig. 5)

- It is necessary to determine, which guidelines and conventions are to be tested using the static analysis tools. The effort undertaken to adapt the tools should not become too large. If need be, one will want to dispense with automated tests for every individual requirements; the requirements that were not tested can then be spot-checked with manual code reviews.
- It is necessary to determine which metrics should be used.
- For every metric employed, it is necessary to define
 - which metrics values are unacceptable (exclusion criteria)
 - which metrics values can be accepted offhand.

All other values that are neither unacceptable nor acceptable offhand signalises that the code should be checked manually by reviewing code.

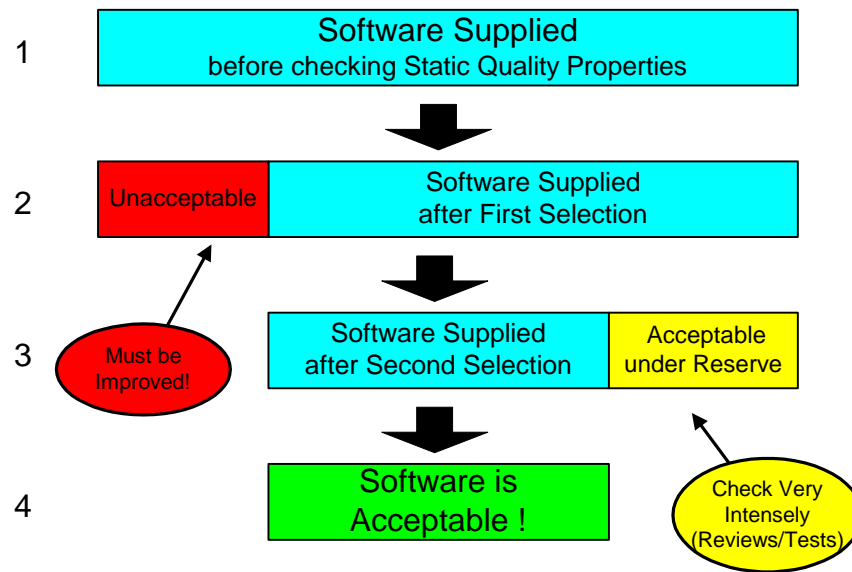


Figure UHEHN.5: Filtering unacceptable or not necessarily acceptable software

After this preparation, the static tests can be conducted automatically. If all static tests are successful and all calculated metric values are acceptable, the software delivered by the supplier is accepted; otherwise, it is returned to the supplier for remedy.

It is recommended to agree with the supplier about how the acceptance criteria are determined early on, i.e. before starting the tests. Ideally, the acceptance criteria should be agreed upon at the beginning of the project; this way, the requirements on the software are clear from the beginning.

Extent of Static Tests

The automated tests should include all the delivered software sources, since this requires only little effort and costs. Since only tested and accepted software is passed on, the share of "not necessarily acceptable" software will not be very high. This part of the software should be subjected to a more intensive code review "from the customer's point of view" (this can also be done by a test lab).

The formal aspects - adherence to the structure template, existence of a version history, etc.- of respective documents, release notes, etc. should also be checked with simple tools.

Outlook

The quality of "passed-on software" can be assured by a combination of several methods. The influence of the customer on the software supplier's quality assurance methods must be strong enough for these measures to show the expected results. Automated tests using static analysis and metrics were suggested to supplement the

software supplier's measures and to provide additional security.

Putting this approach into practice requires that the customer and the supplier agree upon which measures are to be undertaken and that these agreements are lived in everyday business. Considering the product quality and the importance for the end user, the necessity for suitable measurements is essential.

The problem was shown from the customer's point of view. Nevertheless, the presented approach does not only yield advantages for the customer:

- It is very useful for the contractor when the software passed on to him is in a good state. Very often, contractors work under strict time constraints; clearly, problems with the software they must use do nothing to improve this situation.
- The increased influence the customer has on the software supplier might at first be unusual and undesired for the latter; on the other hand, the increased software quality this brings about also strengthens the position of the software supplier, not to speak of the reduced effort and costs needed to process usually high-priority problem reports caused by discrepancies during the integration process at the supplier's or customer's.
- The very fact that someone will look into the software produced will help to make the software better!

References

- [1] Messnarz R. (Ed.): "Better Software Practice For Business Benefit", IEEE Computer Society, 1999
- [2] Koomen T., Pol M.: "Test Process Improvement", Addison-Wesley, 1999
- [3] Hehn U., Haworth B.: "Testing Distributed Embedded Systems; Factors and Solutions", Test Congress 2000, London, May 8-12
- [4] Frühauf K., Ludewig J., Sandmayr H.: "Software-Prüfung - Eine Anleitung zum Test und zur Inspektion", vdf Zürich, 2000

Authors

Dr. Uwe Hehn, born in 1954, studied computer science and mathematics at the Technical University of Darmstadt (Germany). He acquired his doctorate at the University of Erlangen-Nürnberg (Germany). Dr. Hehn worked as developer und project leader in software development and test. In 1997 Dr. Hehn joined 3SOFT GmbH. Since 2001 he is working with method park Software AG, which is a spin-off of 3SOFT GmbH. His work focuses on consulting, training and teaching in the subject of the definition and implementation of test processes and quality measures at customers development and test divisions. Dr. Hehn is guest lecturer at the University of Erlangen-Nürnberg. He published several papers on testing and quality management.

Dr. Bernd Hindel, born in 1960, studied computer science in Erlangen and Green Bay (USA). After his doctorate at the University of Erlangen in 1991, he moved to the Central Research and Development Department of Siemens in Erlangen. In 1995 Dr. Hindel became Managing Director of 3SOFT, where he built up the business

unit Software Process Improvement. Since 2001 he is CEO of method park Software AG. He also works as a Visiting Assistant Professor on Software Development Processes at the University of Erlangen and the University of Würzburg. Since 1996 Dr. Hindel is a member of the board of the ASQF e.V. (Working Group on Software Quality Management www.ASQF.de). He is also a member of several programme committees of international software conferences and published papers on software metrics, testing and project management.

Company

The software company method park Software AG, Erlangen (Germany) was founded in 2001 as a spin-off of 3SOFT GmbH, Erlangen, based on the departments Software Process Improvement and Object Oriented Technology.

Our expertise includes

- design and development of object oriented software
 - internet technology
 - consulting and training related to the software development process and the test process (Software Process Improvement)
- performing seminars and workshops (i.e. about object oriented analysis and design with UML, management of software development processes, definition and execution of test processes, internet technology)

Test Execution and Test Management for Numerical Control Software

Thomas Bürger

FISW Steuerungstechnik GmbH, Stuttgart, Germany

Joachim Mayer

Industrielle Steuerungstechnik GmbH, Stuttgart, Germany

Introduction

This paper points out that test execution and test management are essentially important for Quality Assurance (QA) purposes. Within the IST-funded project TEAM (Test Execution and Test Management for Numerical Control Software) this is demonstrated by the example of an open and modular control system of the company Industrielle Steuerungstechnik GmbH (ISG). This so-called "Numerical Control" (NC) is used for machine tools and manufacturing units.

Continuous software process improvement is a key issue for all software developing companies. As the software part in the field of machine tools and manufacturing units is permanently growing, the software process improvement becomes a topic of increasing importance. This also involves the test process improvement, which is the goal of the TEAM project. This goal comprises the improvement of the NC test execution and the improvement of the NC test management

This paper presents the procedure, methods and tools, which are necessary to improve the test process. An appropriate test procedure is derived from the requirement of the specific NC software development process. The methods for realising this test procedure within the software development process are defined. The tools which have to be used by these methods are described. The finally presented test environment for an on-line NC-Software test demonstrates a tool that allows a fast and effective greybox test for a regression test procedure.

Numerical Control

Employment and Definition of an NC-System

The goal of ISG is to support machine tool builders and control vendors with an NC in a way that they can spend their own innovative forces in their own fields of competence. The NC is used in most of the diverse automated systems, e.g. for 3- and 5-axis milling of wood-working and metal-working, for lathes, for grinding machines, for parallel strut machines, for high-speed plotter and for surgery robots. The task of the NC within such automated systems is to co-ordinate the motion of several driven axes in a way that the position and orientation of a tool relatively to a fixed co-ordinate system (e.g. a work piece) corresponds to predefined values (e.g. by a part program or by joystick command values).

In addition to an NC an automated system contains several software and hardware components which need to be considered in the context of test execution for an NC (Fig. TBURJMAY.1). Therefore the NC has interfaces

- to a Human Machine Interface (HMI) for interactions with the user of the system,
- to a Programmable Logic Control (PLC), which is necessary for logic operations (e.g. safety surveillance),
- directly to process related measured values (e.g. forces) and to the
- drive system, which finally executes the commanded positions and measures the current positions of the physical axes.

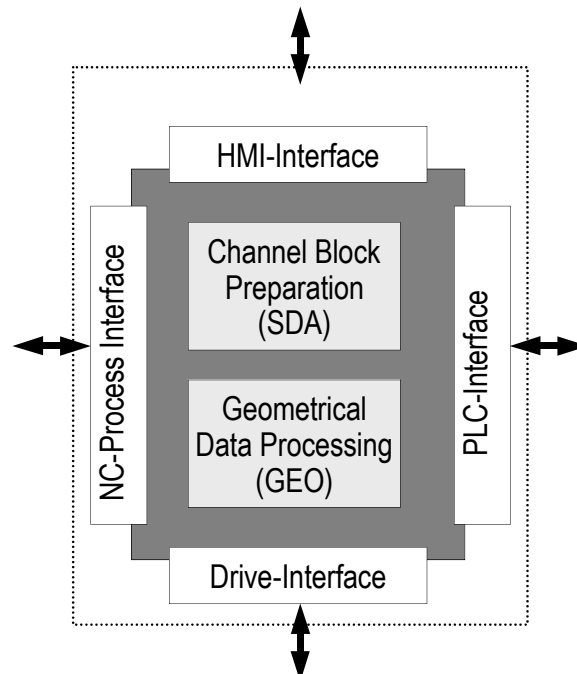


Fig. TBURJMAY.1: NC interfaces within an automated system

The NC system can be sub-divided into further internal software components which run

in different time levels:

- The Channel Block Preparation (SDA), necessary to transform the geometrical data (defined by a user e.g. by CAD) to an NC internal format, which runs in a non real-time critical mode and
- the Geometrical Data Processing (GEO), necessary to transform the NC internal data format to command values for the drives, which runs in a separate real-time task (usual cycle time ~2 msec).

Both components, the Channel Block Preparation as well as the Geometrical Data Processing, comprise further software modules which are necessary for different functional purposes (e.g. tool radius compensation, co-ordinate transformation, etc.).

NC Software Development Process

The constantly increasing complexity of machine tools and manufacturing units result in a continuously growing demand on the functionality of NC. The resulting increase of functionality of NCs is mainly realised by functional extensions of the software of an NC. Due to the large scope of the already existing basic functionality of a modern NC software (about 200 person-years of development effort), the set-up of a new NC software version is based on the existing basic functionality of previous versions. Therefore the development process of NC software can be described as a further incremental development of an existing system (Fig. TBURJMAY.2) [1].

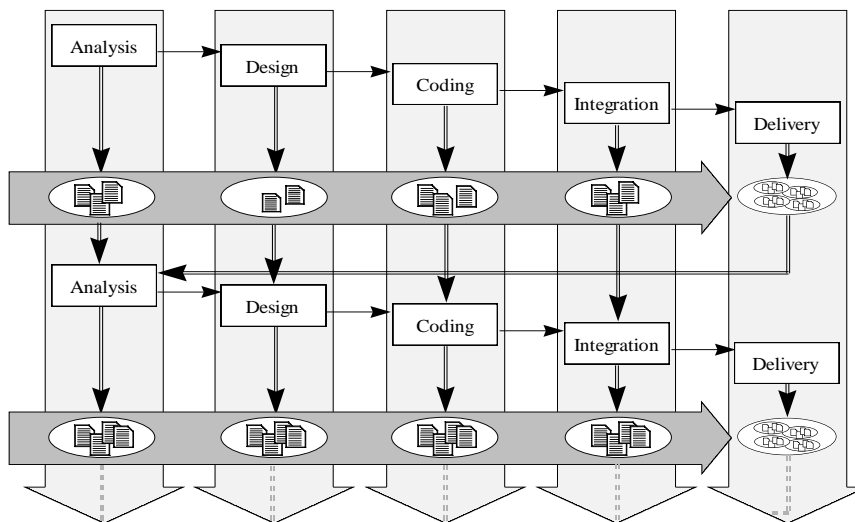


Fig. TBURJMAY.2: Further incremental development of NC software

Change Processes and relating Test Methods

Change processes do not only occur due to functional extensions but also due to functional improvements and bug removal activities. At ISG those activities are executed parallelly by 22 software developers at one NC software system. In order to enable this further incremental development, there exist several customer specific variants (branches) of the NC software system.

The change processes executed in order to set up a new NC-Software version can be

distinguished into three major change processes (Fig. TBURJMAY.3).

The most extensive change process (CP1) contains all important phases of a software development and is usually executed on demand of a customer in order to realise a new and complex functionality within the NC-Software. By this, e.g. a new manufacturing technology is supported. The second type of change process (CP2) is executed as soon as a change request (CR) or an error message (EM) is released by the development manager. CR and EM can be applied and reported internally as well as externally by a customer. The third type of change process (CP3) describes the typical way of software development that is executed under pressure of time. The coding is initiated after short discussions without specifying the relevant software changes in detail.

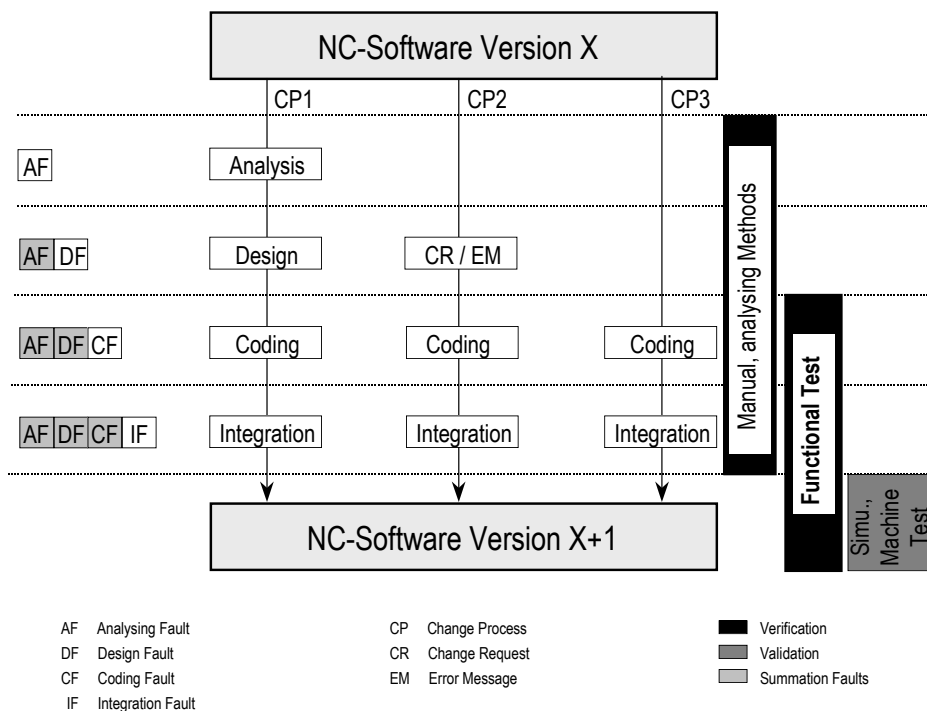


Fig. TBURJMAY.3: Change processes for the development of NC software

Executing manual, analysing methods (e.g. reviews, walkthroughs or inspections) are time consuming and require motivated and high skilled examiners. For the validation of NC-Software by the use of simulations or machine tests [2], experienced NC developers are necessary to judge the correctness of the system behaviour. Both the verification of NC-Software with manual, analysing methods and the validation of NC-Software are disadvantageous due to the fact that the correctness of the test result (document or system behaviour) has to be proven manually.

The main requirement for the test execution is to run this test in an effective way. The focus of the test process improvement is therefore put on the NC-Software verification by the automatic execution of so-called "functional tests".

As nevertheless the manual, analysing methods and the validation of NC-Software by simulations and machine tests are important within the NC-Software development, the application of these test methods has to be optimised within a general test management improvement.

Execution and Goals of the TEAM Project

For the execution of the IST-funded project TEAM, the V-Model [3] functions as a supporting guideline. The V-Model is a Lifecycle Process Model and was originally developed as a standard for the Federal Administration of Germany. Today it is also used as a standard for a great number of software and system houses. The main focus of the QA submodel of the V-Model is on the introduction of analytical QA activities to the software development process. This analytical QA activities comprise verification and validation activities. The introduction of suitable analytical QA activities, which are necessary to detect and localise weaknesses and defects within the software components, is important for improving the test process. So the goal of the TEAM project is a test process that is effective from a cost (time and resources) benefit (quality improvement) point of view.

For all its submodels (besides QA there are System Development (SD), Configuration Management (CM) and Project Management (PM)), the V-Model establishes three levels:

- Procedure – "What has to be done?"
- Methods – "How is something to be done?"
- Tool Requirements – "What has to be used to do something?"

According to these levels of the V-Model, the objectives for the execution of the TEAM project are defined (Fig. TBURJMAY.4).

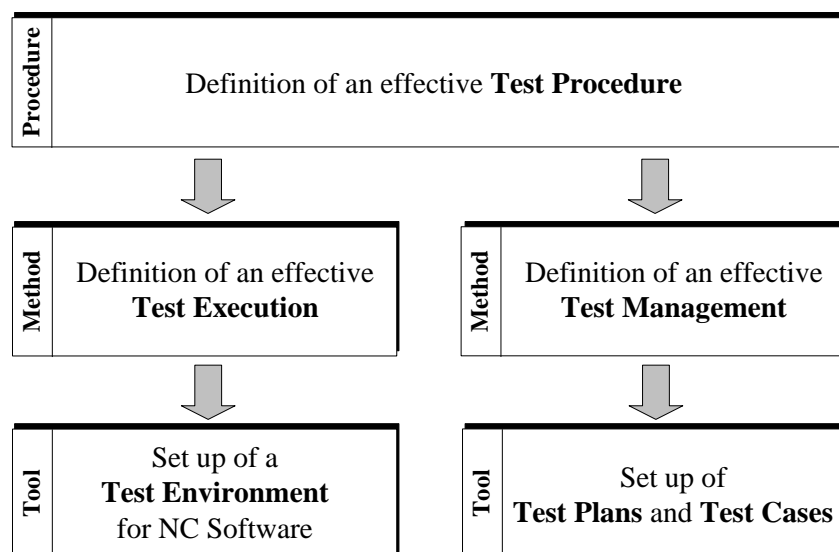
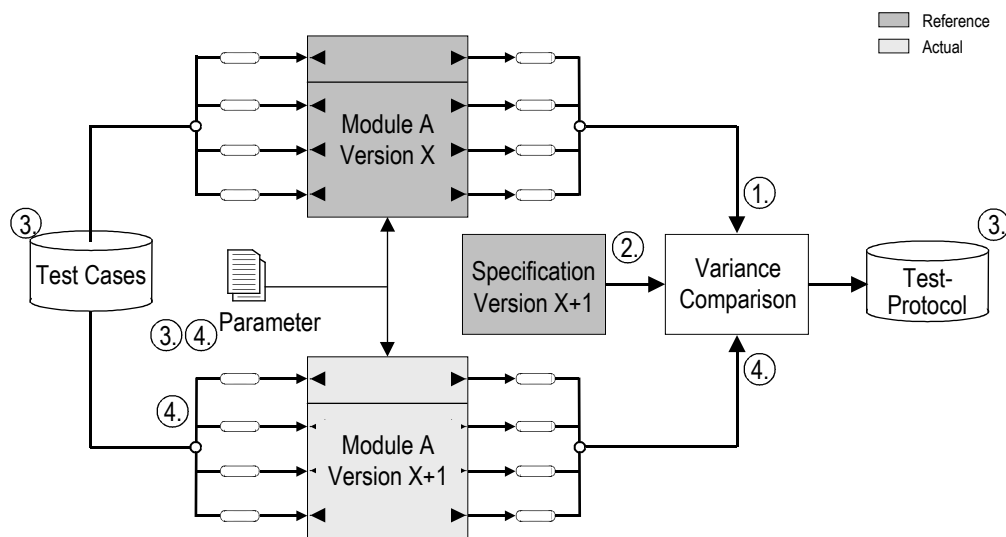


Fig. TBURJMAY.4: Objectives of the TEAM Project

After the definition of an appropriate test procedure, the methods for an effective test execution and an effective test management have to be defined. For these methods, the respective "tools" have to be set up.

NC-Software Test Procedure

Examinations within the European funded InCoMM project executed at ISG pointed out that about 75 % of all registered bugs within the NC-Software arise during coding and integration work [4]. Functional tests have to be executed in order to discover these bugs emerging within the development phases and to check the functional completeness [5] of the NC-Software after the completion of a change process. As the NC-Software development process is a continuous further development of an existing system, and as the development therefore is the change of already existing NC-Software, regression tests are particularly suitable for the execution of functional tests (Fig. TBURJMAY.5).



- | | |
|----------------------------------------------|----------------------------------------|
| ① Test of basic Functionality (Verification) | ③ Reproducible |
| ② Test of new Functionality (Verification) | ④ Automated (Execution and Evaluation) |

Fig. TBURJMAY.5: Regression Test for Functional Test

Test procedures comprise always a comparison of desired results and actual results. The desired results can be described in different ways (e.g. formal description). Within a regression test the desired results can be generated automatically by a diverse existing version that was already verified and validated by former tests and by the employment in machine tests.

For the input (test cases) of the regression test, different categories can be applied:

- NC part program
- Operator action (via HMI)
- PLC action

For the output of the test, which is recorded after a variance comparison in a test protocol, the following categories can be considered:

- NC axis movement
- Signals from the NC to the PLC
- Protocol data of NC internal data (e.g. function blocks)

The mentioned "Module" under test (Fig. TBURJMAY.5) can either be a complete NC system, an NC component (SDA, GEO) or software modules, which are part of these components. For the functional regression test of specific modules of an NC, NC part programs are used as an input:

First, the test cases of the respective test run and the respective test configuration are selected from a test data base and compiled to a test script. The test cases are formulated as NC programs (e.g. according to DIN 66025) in the same way they are used in the production with NC machines. In the subsequent test run these test cases (NC programs) are sequentially processed by the NC. At the same time, the NC generates function block protocols and test run protocols (logfiles). Function blocks describe the internal data format (process data and control data) of an NC and represent the test output of the respective test case within a functional test.

The test version is configured and parameterised identically to the reference version. This enables to compare the function block protocols of the test version with reference protocols of an already tested version. Because of this and due to the use of a standardised protocol routine, it can be guaranteed that all differences identified during the variance comparison of the actual protocols with the reference protocols are due to changes within the functionality of the NC software. Those differences identified can be explained either by planned changes in the source code or by bugs.

NC-Software Test Methods

For the execution of the functional test, two different methods are proposed within the literature [6]:

- Blackbox-Test, where the tester views the NC-Software as a black box only from its interfaces (Fig. TBURJMAY.1) and the test is completely unconcerned about the internal structure of the software.
- Whitebox Test, where the tester views the internal structure of the NC software and tries to achieve a high test coverage, that is examination of as much of the statements, branches and paths as possible.

The internal structure of the NC-Software has a sub-division in different modules which are necessary to support different functionalities. As the interfaces between these modules are precisely specified and standardised, a pure blackbox test for the execution of the functional test would ignore the information recorded and evaluated at this NC internal interfaces. The execution of whitebox tests, which have the goal to achieve a high test coverage, is not practicable, as the NC-Software is too complex and extensive. Therefore a so-called "greybox test", examining the NC internal interfaces, is considered to be suitable for the NC-Software functional test purpose.

According to the time of the execution of the variance comparison (Fig. TBURJMAY.5) two different methods can be distinguished within a regression test:

- Off-line comparison, where the outputs of the reference version and the actual version are separately recorded on disk before the comparison.
- On-line comparison, where the outputs of the reference version and the actual version are generated simultaneously and are compared continuously while the test is running. Only detected differences are written to a test protocol.

Considering the specific requirements of the NC-Software (internal structure) and of the NC-Software development process shows that an on-line greybox test for a regression

test procedure is the most effective way for test execution.

With the availability of an effective method for test execution the necessity of an effective test management arises in order to apply an effective test procedure to the NC-Software development process. For this reason the existing change processes and the applicability of the individual verification (e.g. the mentioned on-line greybox method) and validation (e.g. machine tests) methods for this change processes have to be examined. This examination is currently done within the TEAM project.

In order to apply the tests into the software development process in an effective way, it is important to examine which verification or validation method has to be executed at which point of time and for which change process (CP). The execution of functional tests depends e.g. on the following categories:

- The module(s) of the NC-Software affected by the change.
- The manual, analysing methods executed before (depending on the CP).
- The last version of the NC-Software that can be used as reference.
- The subsequently following analytical QA activities that are planned (e.g. machine test).

NC-Software Test Environment

For an effective execution of the above mentioned greybox test, a regression test with an on-line variance comparison is required. The TEAM project therefore intends to set up a test environment for the execution of such an on-line NC-Software test (Fig. TBURJMAY.6).

All relevant input categories (NC part program, operator action or PLC action) that are defined as test cases shall be executed by the actual version of the NC-Software and by the reference version of the NC-Software in parallel. Particular synchronisation mechanisms are to be realised in order to gain comparable results. The results will be recorded from internal (between NC modules) and external interfaces (to other components within an automated system) of the NC-Software. An on-line variance comparison of the results reduces the size of the test protocols that have to be stored. Furthermore, the evaluation of the overall test result can be reduced to the evaluation of the differences between the actual version and the reference version of the NC-Software.

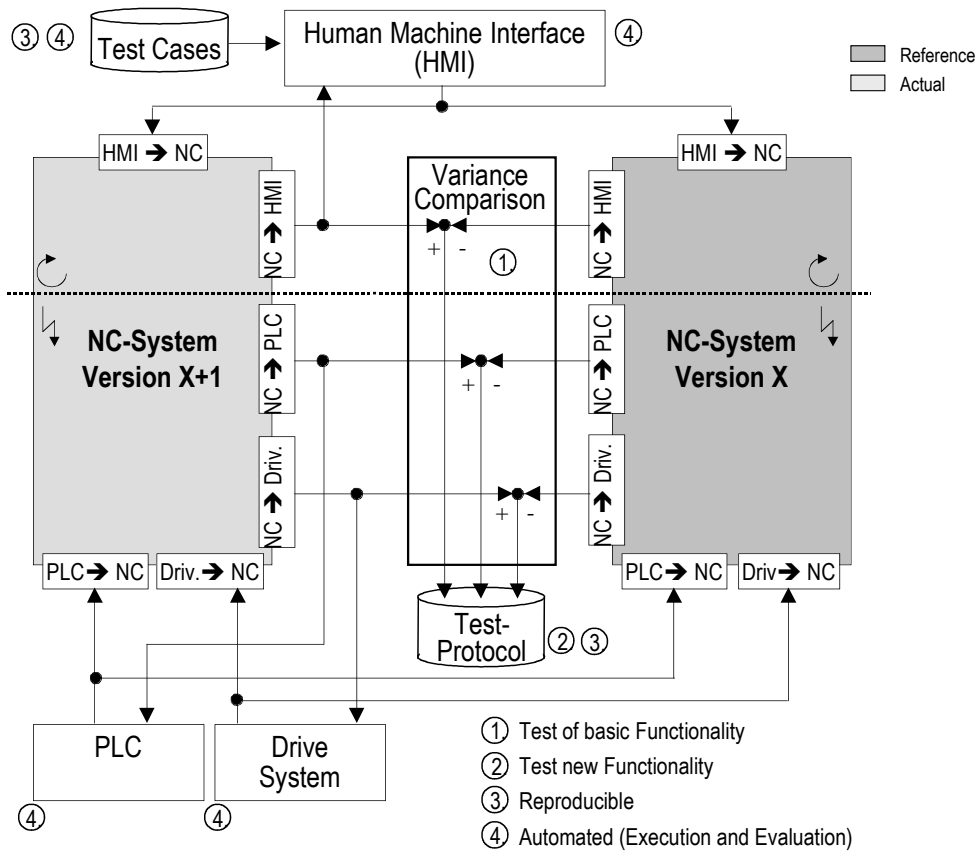


Fig. TBURJMAY.6: Test Environment for an On-line NC-Software Test

An effective test management bases on precisely defined test plans and test cases. Within the TEAM project one focus will be on the integration of the already existing field of CM into the test management. The test plans, test cases and test results will be considered as software elements that have to be managed (version control and change management) by the CM. During the execution of test plan instructions the version control of the NC-Software will be necessary in order to provide the correct NC-Software for the verification and validation methods.

Considering for example the execution of functional regression test with NC part programs for input purpose, the necessity of an effective test management is emphasised. In order to provide the appropriate versions of the NC-Software (reference and actual) and the correct test cases to the test environment an effective CM is required [1, 7]. This is also necessary for recording the test results (test protocol) in a way that all tests are reproducible at any time.

Conclusion

A successful and effective software development requires an effective test procedure with appropriate methods and tools. Due to the specific requirements of the NC-Software development process, greybox tests, executed as regression tests, are particularly suitable for the execution of functional tests. In order to execute this kind of test in an effective way, a test environment will be set up within the TEAM project that provides an on-line evaluation of test results.

This effective way of testing NC-Software will be supported by a test management that uses the services of CM. The QA methods mentioned within the test plans will be applied to NC-Software elements which are under control of the CM. This guarantees a high effectiveness and reproducibility of the test executions.

The knowledge gained out of the TEAM project concerning test execution and test management, contributes to an improved test process at ISG and therefore to a continuously high quality of the NC-Software too. In order to offer NC-Software for new applications on the market (e.g. for medical devices or for training machines used at vocational schools), ISG needs to prove the quality and safety of its NC to a notified body. This requires reproducible and appropriate test executions. The positive results of the TEAM-project will support ISG's efforts to open up these new markets.

References

- [1] Beeh R., Bürger T., Quality Assurance for NC-Software with the support of Configuration Management, in: *Proceedings of the EuroSPI 99 Conference*, pp. 9.12-9.23, ISCN, Pori, Finland, 1999.
- [2] Meier H., Kreuzsch K., CNC-Test an virtuellen Werkzeugmaschinen, in: *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, pp. 415-417, Carl Hanser Verlag, München, Germany, September 1998.
- [3] Entwicklungsstandard für IT-Systeme des Bundes - Vorgehensmodell, Teil 1: Regelungsteil, Allgemeiner Umdruck Nr. 250/1, Oldenbourg Verlag, München, Wien, 1998.
- [4] Pritschow G., Beeh R., Bürger T., Qualitätssicherung in der CNC-Steuerungsentwicklung, in: *Zeitschrift für wirtschaftlichen Fabrikbetrieb*, pp. 353-356, Carl Hanser Verlag, München, Germany, June 1999.
- [5] Liggesmeyer P., Rothfelder M., Ackermann T., Qualitätssicherung Software-basierter technischer Systeme - Problembereiche und Lösungsansätze, in: *Informatikspektrum*, pp. 249-258, Springer Verlag, Berlin, Heidelberg, Germany, October 1998.
- [6] Myers G.J., Methodisches Testen von Programmen, Oldenbourg Verlag, München, Wien, Germany, 1999.
- [7] Scheifele D., Beeh R., Bürger T., Nagel T., Configuration Management for Open Modular Control Systems, in: *Proceedings of the Sixth European Conference on Software Quality*, pp. 526-534, Arbeitsgemeinschaft für Datenverarbeitung (ADV) mbH, Wien, Austria, April 1999.

Glossary

CAD	Computer Aided Design
CM	Configuration Management
CP _x	Change Process No. X
CR	Change Request
EM	Error Message
GEO	Geometriedaten Verarbeitung (Geometrical Data Processing)
IST	Information Society Technology
NC	Numerical Control
PM	Project Management
QA	Quality Assurance
SD	System Development
SDA	Steuerdaten Aufbereitung (Channel Block Preparation)
TEAM	Test Execution and Test Management for Numerical Control Software

Appendices

About the Authors

Thomas Bürger was born in 1970. He studied mechanical engineering at the University of Stuttgart. Since 1996 he has worked as a research assistant at FISW Steuerungstechnik GmbH in the field of machine-oriented control functions.

Joachim Mayer was born in 1967. He had been an employee of Mercedes-Benz AG from 1984 to 1989. Later he studied mechanical engineering at the Fachhochschule Reutlingen (from 1989 to 1993). From 1993 to 1999 he had been employed as a manufacturer of electrical discharge machine tools, where he was responsible for CNC-control software development, mechanical and electrical development. Since 1999 he has been responsibly working in the fields of software development and quality assurance at Industrielle Steuerungstechnik GmbH.

About FISW Steuerungstechnik GmbH

FISW Steuerungstechnik GmbH is experienced and qualified in the areas of control technology, co-operation and co-ordination of research areas of the Deutsche Forschungsgemeinschaft (DFG), co-operation and management of local, national or European joint projects, consultation in industry, and processing development tasks given by the industry in various areas. Beside the public partners of FISW Steuerungstechnik GmbH co-operations with partners out of all fields of industry dealing with software for control technology exist. FISW Steuerungstechnik GmbH main experience relating to the TEAM-project is the experience in quality assurance for NC software.

About ISG

The company Industrielle Steuerungstechnik GmbH (ISG) develops and sells software elements as components for open and modular control systems. The customers of ISG are either control vendors or machine tool builders, who use software of ISG (source code and accompanying documentation) for their own controls. The software of ISG can be extended with their own software modules. This openness results in a great number of software elements such as source code files, documentation, or test patterns.

Being a service provider, ISG places itself at the demands of its customers and employs 22 engineers in the central business field, which is the development and maintenance of NC software. It has an annual turnover of about 4 Mio DM.

ISG was founded in 1987 with the aim to support control vendors and machine tool builders with an open control system in such a way that they can concentrate on their own innovative strength in their fields of competence. ISG's NC software is used in a wide range of technologies. Thus, it is in use for e.g. 3- and 5-axis milling of metal-working and wood-working, for lathes, for grinding, for a various number of technologies with parallel strut machines, for high-speed plotter, robots and robotal devices.

Session 4 - SPI and Development Paradigms

Session Chair:
Mads Christiansen, Delta, Denmark

Managing Projects in a Rapid Application Development (RAD) Environment	4-2
Evolutionary Development of Web-applications – Lessons Learned	4-14

Managing Projects in a Rapid Application Development (RAD) Environment

Dr. Patricia A. McQuaid

California Polytechnic State University

College of Business - MIS

San Luis Obispo, California 93407

United States

email: pmcquaid@calpoly.edu

Abstract

We seem to be continuing our search for what Fred Brooks called the “Silver Bullet” [1]. We have made technological advances, created new tools, followed a process improvement plan, utilized benchmarking, developed measurement programs, performed rapid application development (RAD), and modified our software testing methodologies. However, project management is still not done well. Reasons for this include poor estimation techniques, the lack of historical data, weak management commitment, inadequate or nonexistent processes, planning being done in a vacuum, the project plan and other plans not being synchronized, lack of effective software testing being performed, and re-planning rarely done, even if crucial factors change. This paper surveys many of the important issues that project managers face while managing software projects in such a dynamic environment.

INTRODUCTION

Quality Management is not done well. Part of the reason for this is due to a lack of understanding of what the quality functions are. Quality is still viewed as an overhead function to be done only if the customer pays for it, and too many projects do not realize the value added from quality functions or resources. Even today, measurement data on quality functions is largely ignored. There seems to be an inability to manage the software engineering process: processes are chaotic, there is a lack of engineering discipline, and the quality framework and measurement discipline is missing. In terms of people, the organization structure, skills and culture development, career paths are not clear, motivation systems and compensation are not aligned with business objectives. Too often, organizations are hiring *only* for the project, and not looking out long term for the hiring needs of the organization. There are often no standards to deliver the product in a certain way, resulting in an inconsistent delivery of applying the quality function in organizations.

There are a number of reasons that not only RAD projects fail, but that software projects in general fail. A core reason is due to the lack of project management training by the people responsible for the project, leading to unrealistic expectations and schedules, problems with requirements, and problems with the developers and customers. Software development is difficult to manage even if the requirements could be fully and accurately defined at the beginning of a project. Normally, the requirements are so complex that requirement changes are inevitable during development, with the difficulty compounded when the time frame is short. The challenges of managing requirements and controlling resources tend to result in the familiar panic mode and crisis mode around the time of delivery. There may be an absence of an effective configuration management system. There may have been too little customer involvement. As a result, the quality of the delivered product is poor, rework pressure is increased, and the employees become burned out. To further compound the problem, the testing of the product can suffer greatly. Capers Jones reports that poor quality is one of the most common reasons for cost overruns and is also one of the reasons for close to half of the canceled projects [2]. This paper will define Rapid Application Development (RAD), discuss the reasons for failure in more detail, and provide some recommendations to help alleviate these problems.

WHAT IS RAPID APPLICATION DEVELOPMENT?

Rapid Application Development (RAD) is a term for a project that emphasizes development speed and, if done properly, can be structured and disciplined. Despite the word “rapid”, it is *not* intended as a Quick Fix proposition. RAD concentrates on the delivery of the product and involves the client from the start and focuses on their needs, uses an incremental approach, keeps the project plan updated, applies development

fundamentals, and manages risks to avoid catastrophic setbacks. A major goal is to avoid what Steve McConnell [3] calls the “Classic Mistakes”, discussed later in this paper. RAD is important because “The ability of an information system to evolve to meet new requirements is one of the key quality characteristics, but systems must also be capable of rapid evolution if they are to deliver real value to the business community in this volatile business environment. Systems which cannot evolve rapidly offer little or no support to their users, who in turn become less responsive to their environment and consequently incur increased risk of failure in the market place” [4].

Successful Rapid Development

Successful rapid development starts with understanding and defining your client’s business needs, and then moves through the phases of high level requirements to detailed requirements, to design, to prototyping, to development, and to implementation. Testing should be involved early in the project and throughout the development effort. One of the goals of RAD is to provide an updated “look and feel” of the evolving product and to allow the client to have “hands on” contact with the product as soon as possible.

Throughout these phases, one must continually review and update the project plan as necessary, carefully controlling all change requests along the way. One must assess the risks to the project at the completion of each cycle and review the current understanding of the client’s business needs throughout the project. The schedule must be accurately developed and carefully controlled. Most projects overshoot their estimated schedules by anywhere from 25-100 percent, but some organizations can predict the schedule accurately to within 10 percent, and 5 percent is not unheard of [2].

Components of Rapid Application Development

Figure 1 lists the major components of a RAD project. Depending upon which software development lifecycle you are using, it may or may not look similar to your existing infrastructure. A version of the product is developed, is shown to the customer/client, and the product is refined based on customer feedback.

Analysis of High Level Requirements	Control of Requirements Changes
Project Management	Risk Management
Design (From Detailed Requirements)	Acceptance or User Testing
Prototyping	Deployment
Development	User Training
Unit, Integration, Usability, Acceptance, and Systems Testing	

Figure 1.

The Goals and the Reality of Rapid Application Development

The goals of RAD include those of building in quality, preventing “feature creep”, controlling the schedule, and maintaining a predictable ship date. However, the reality of RAD is that often usability suffers, performance suffers, maintainability suffers, and testing is not adequately done. A successful RAD project must therefore be carefully managed.

FOUR DIMENSIONS OF DEVELOPMENT SPEED

According to Steve McConnell [3], there are four dimensions of development speed: people, process, product, and technology. These can be considered the major determinants of software cost, schedule, and quality performance. There is a point where the focus on these four factors becomes synergistic, as good practices tend to support one another. Associated with these factors are certain classic mistakes, those ineffective development practices that have been chosen so often, by so many people, with predictably poor results. If these mistakes are avoided, you may not be guaranteed rapid development, but if you do not avoid them, you will certainly be developing at a slower pace. First the four factors will be described, and then some of the most significant classic mistakes associated with the factors will be discussed.

Dimension One – PEOPLE. Why Such a Concern About People?

Software Development is large scale, integrated, intellectual work. Knowledge is the raw material of software development, and it is software engineers who transform the knowledge into software products. Software is the most knowledge-intensive industry in history, therefore organizations must find ways to increase the knowledge, skill, and performance of its software developers to increase the capability of their workforce. Doing so will help the organization to stay competitive in a global market, satisfy the exponential growth of software volume and complexity, and increase the quality and reliability of software systems to levels achieved by hardware.

To achieve the concepts in RAD, or any development methodology, an organization may need to change the way they manage their people in order to improve the skills of the workforce, to develop proud and satisfied employees, to retain corporate knowledge, and to develop competitive people. Teams can be formed in order to create and maintain competitive organizations. We need to ensure that the software development capability is an attribute of the organization rather than the presence of a few extraordinary individuals or heroes. We need to align the motivations of the individuals with that of the organization, and retain the “human capital” and growth so that it is unmistakably a

corporate asset.

Dimension Two – PROCESS. Why Focus on Process Improvement?

An underlying goal of process improvement is to remove defects early and efficiently from the software work products. One of the many benefits of focusing on process improvement (both technical and management methodologies) is that individuals develop the necessary skills and knowledge in order to perform their roles effectively and efficiently. This knowledge improves the understanding of how the organization develops software, helping to increase continuity. It is critical for the developers involved in the software project to establish a common understanding of the customer's needs, not just the requirements, and to keep the customer involved in the development process.

One must establish reasonable plans for performing the software engineering and for managing the software project. It is important to provide management with adequate visibility into the processes being used by the software project, the actual progress of the project, and of the resulting quality of the products being built. An important part of measuring the progress of the project is estimation: you should estimate the size of the product, estimate the effort, and estimate the schedule. Estimating the size of the product is one of the most difficult aspects and many organizations use function points for this type of estimation. The effort estimation is not as difficult, but only if you have historical data on similar projects performed by your organization. Once these estimates are calculated, it is then possible to estimate your schedule. It is helpful to provide estimates in ranges and associate probabilities with these ranges. Once these estimates are made, it is very important to monitor and continually re-assess what you have planned to control your development schedule.

Improving software development processes will lead to reduced rework. This reduction in rework leads to reduced development cycle time, giving increased predictability and control of software quality and a reduction in the number of defects. Another benefit of process improvement is that it can provide increased control of costs and therefore the ability to *predict* both the development cycle length and costs. This enhanced ability allows one to make cost-benefit tradeoffs of development methodologies, technologies, and processes. This level of control increases the ability to make risk management decisions based on quantitative data. One must also be sure to select and manage qualified subcontractors, because their success or failure could dramatically affect your project. All of the benefits discussed result in increased productivity, rather than on dealing with crises.

Process Models – Capability Maturity Model Overview

The Capability Maturity Model[®](CMM), a management-level framework for understanding, managing, and improving software development, is summarized in Figure

2 below [5] and refers to version 1.1 of the CMM. The CMM is a model for organizational software improvement, and is an underlying structure for reliable and consistent software development. The software CMM addresses widespread problems in managing software projects, including quality and productivity issues. By using professional judgment, the CMM can be used across a wide range of size, application domain, lifecycle, development, and maintenance environments. The hierarchical structure of the CMM supports variations by rating processes at the key process area (KPA) goal levels and by using the KPAs, subpractices, and examples as guidance for improving software processes. Factors that must be taken into consideration when an organization is trying to establish its organizational processes can be found in [6]. (The Capability Maturity Model®(CMM), Personal Software Process®(PSP), Team Software Process® (TSP) are registered trademarks of the SEI.)

Level	Process Characteristics	Key Process Areas
5 Optimizing	Process improvement is institutionalized	Process Change Management Technology Change Management Defect Prevention
4 Managed	Product and process are quantitatively controlled	Software Quality Management Quantitative Process Management
3 Defined	Technical practices are integrated with management practices and institutionalized	Process Focus Training Program Peer Reviews Integrated Software Management Process Definition Software Product Engineering Intergroup Coordination
2 Repeatable	Project management practices are institutionalized	Requirements Management Configuration Management Project Tracking and Oversight Subcontract Management Project Planning Quality Assurance
1 Initial	Process is informal and ad hoc	

Figure 2.

Dimension Three – PRODUCT. What Should Be Controlled?

Focusing on the product size and characteristics presents great opportunities for schedule reduction. The schedule may be able to be shortened by reducing the product's feature set. In fact, the 80/20 rule may apply here: perhaps you can develop the 80% of the product that takes 20% of the time. If the feature set is flexible, you may be able to keep the look and feel of the product, its performance and quality characteristics flexible, and construct the product by maximizing the use of pre-existing code. One of the largest contributors to the development schedule is the product size, so by striving to develop the most essential features or developing the product in stages, you can reduce the

development time. Developing the product in stages may be a strategy to deliver components of the software, so that the user may start taking advantage of the product earlier. Defining and implementing the proper characteristics of the software product are critical success factors. Additionally, development effort may be improved by minimizing other product requirements, such as overly ambitious goals concerning performance, robustness, reliability, portability, and so on. As you can see, it is critical to involve the customer early and continually in the development process, which is one of the key RAD components. Negotiation skills are an important aspect of controlling the schedule and feature set.

Dimension Four – TECHNOLOGY. What Effect Does Technology Have?

Development speed can be improved by moving from less effective to more effective tools. Choosing tools effectively and managing the associated risks is a key technique to achieving RAD. An efficient implementation strategy is a key technique to achieving RAD. One needs to be very careful not to believe that it will be the new technology that will save the day.

CRITICAL MISTAKES

As noted earlier in the paper, McConnell [3] describes certain “Classic Mistakes”, those ineffective development practices that have been chosen all too often, by many people, with predictably poor results. The goal is to avoid these mistakes so that you reduce the risk of slow development and achieve rapid development. The author has selected a subset of these mistakes for each category, those deemed to be the most important. The remainder of this section deals with these problems and proposes some recommendations.

People-Related Mistakes

Work done by Tom Demarco and Tim Lister [7], among others, have shown that people issues have more impact on both the productivity and quality issues than any other factor. Oftentimes, the organizational infrastructure is not properly in place to support the development and management of highly skilled and motivated people. All too often project managers are trained on technical matters, and not very much on the management of people. As a result, these kinds of managers may generate high employee turnover. Oftentimes, if a project is in a crisis, people may be added to late to the project in the hopes of meeting the deadline. However, it may take 6 months or more to have a productive employee [3]. To further add to the problem, heroics are often rewarded publicly and true successes are often left unnoticed, leading to an enormous level of

frustration by all employees involved in the project.

For many years, software developers and engineers have worked on their own, writing their programs and developing their software. But now, times have changed and teamwork is highly valued. The ability of people to work in teams is another skill, which while is intuitive to some, must be learned by others.

People-Related Recommendations

It is critical that project managers be trained to effectively manage their employees. To provide guidance on these issues, Watts Humphrey of the Software Engineering Institute (SEI) has developed the Personal Software Process[®] (PSP) [8], a software process that is based on quality management principles to be used by an individual software engineer to help manage and improve performance. With PSP, engineers use process management principles, measuring and analyzing their processes to improve the accuracy of their plans and the quality of the software they produce. The PSP Quality Strategy stresses that low defect content is an essential prerequisite to a quality software process. The personal level is where defects are injected and where the engineers should remove them, determine their causes, and learn to prevent them. There is also ample material related to managing software developers available from practitioners in the field, such as [7] and [3].

More and more projects are being undertaken in teams and the importance of group dynamics for those developers working in teams cannot be over emphasized. To provide guidance to the problem of working in teams, Watts Humphrey has also developed the Team Software Process (TSP) [9], which extends and refines the Capability Maturity Model (CMM) and Personal Software Process (PSP) methods. It was developed to help software engineering teams build quality products within cost and schedule constraints and to accelerate software process improvement. It provides guidance to organizations in how to build a self-directed team and how to perform as an effective team member in both development and maintenance work. It also shows management how to guide and support these teams and how to maintain an environment that fosters high team performance. Consultants Demarco and Lister have also published a number of books and articles related to managing teams. Fortunately, many universities are incorporating a great deal of teamwork into software development, software testing, and project management classes in both business and engineering curriculums.

Process-Related Mistakes

Process-related mistakes hamper development by wasting the time and effort of the developers. A central problem is having poorly defined software development processes, procedures, plans, guidelines, and templates.

Very often, projects get off to a bad start because scheduling is done by upper-level management and/or marketing, resulting in overly optimistic schedules. Even when the schedule is established by software developers in a well-defined manner, it is not uncommon for others to modify the schedule to fit their needs, thereby increasing the risk of failure. It is critical to identify, address, and eliminate sources of risk before they

become threats to your project, so organizations must have an adequate risk management focus. Some organizations have the “Code-Ship-Test” mentality. Having an inadequate understanding of the necessary quality functions throughout the lifecycle clearly results in the production of bad software. Too often there are insufficient management controls and oversight into the processes being used on the projects and their ability to produce the necessary product quality.

Without a strong configuration management process, problems can occur, such as the following: the latest version of source code cannot be found, a difficult bug that was fixed at great expense suddenly reappears, a developed and tested feature is mysteriously missing, or the wrong version of the code was tested. Finally, a fatal error is for an organization to abandon planning and most other processes when under pressure, when controls are most needed.

Process-Related Recommendations

Quality needs to be designed in from the start, and not short-changed during the development process. Quality assurance (QA) fundamentals must be in place, including the development of a Quality Plan. QA’s role is to ensure that the project is developed in compliance with the defined processes and that the processes defined are adequate for the project. It also provides feedback to the project’s management about the effectiveness of the processes the developers are following and provides feedback to the process group about the usability of their processes.

Configuration Management is the backbone of the development process and helps to ensure product quality and process improvement. Further information on configuration management and its impact on project management can be found in [10]. It involves identifying the configuration of the product (i.e., selected work products and their descriptions) at given points in time, and systematically controlling changes to the configuration. It helps to maintain the integrity and traceability of the configuration throughout the life cycle. It is a process that manages product evolution throughout its life cycle and creates a verifiable history of the product as it matures. It enhances communication among project members and customers.

As part of the development process, not only must an organization put in place a risk management plan, but it needs to become a formal part of the development process. Schedules must be realistically set so that the scope of the project is properly defined, planning can be effectively done, and the developer morale and productivity can be maintained. Requirements must be managed properly. When you see that the project is falling behind schedule, do not abandon planning and “do whatever it takes” to finish the project, but take the necessary time to re-plan. One must adhere to their software configuration management plan to establish and maintain the integrity of the products of the software project throughout the project’s software lifecycle. A testing methodology must be emphasized, including unit testing, integration testing, systems testing, usability testing, user acceptance testing, and regression testing throughout the process.

Product-Related Mistakes

Since one of the largest contributors to the development schedule is the product size, managing aspects affecting the size is critical to a successful development process. One common mistake is that of requirements gold-plating, those requirements included by the customer that are not essential. Related to this is “feature creep”, those requirements that keep changing throughout the life of the project. Gold plating need not be done by just the customer, but can also be done by the developer, by adding features that were not requested by the customer, but that the developer feels the software needs. Clearly, inadequate control of requirement change requests can be fatal to a project. Not specifying acceptance criteria by the customer can be a problem. An often overlooked consideration is assuming that if the product meets the specifications, it will be viewed as quality by the client. So, once again, for the reasons described here, it is important to continually work with the customer.

Product-Related Recommendations

Gold-plating by either the customer or the developer needs to be carefully controlled, as does feature creep. Perhaps some of the desired features can be included in the next release, not the current release. If some feature needs to be included, then perhaps you can negotiate and defer a feature that is in the current plan or extend the deadline. All parties involved must be made aware of the effects to the schedule when requirements are modified and that the risk of missing the deadline is increased. Controlling requirements change requests is crucial to the success of the project, as is a strong configuration management program. Changes to requirements must be well documented, approved by all necessary parties, and tracked. Customer acceptance criteria must be defined, because how else do you know when you are done? It is very important here to involve all stakeholders in the requirement gathering and monitoring phases. A common mistake is to not identify all the stakeholders in the beginning of the project. Not only should you involve the users, developers, and customers, but do not forget to involve your test teams and network personnel.

Technology-Related Mistakes

All too often, project teams search for the Silver Bullet, relying on new technology to save the day. Perhaps the new language or the new hardware or the new tool will solve the schedule problems. Or perhaps the savings attributed to the new tool or method has been overestimated. There may be a lack of understanding that when a new tool or technology is introduced to a project, productivity will first go down before it goes back up, due to the learning curve involved. Introducing the new technology to the entire organization too fast can cause increased delays in the schedule. Another problem is switching tools in the middle of a project. Between the learning curve and the rework involved, the benefits of the new tool can often be cancelled out.

Technology-Related Recommendations

Rather than relying on a new technology to save the project, emphasis should be placed on solid development principles, as described throughout this paper. It is wise to phase in the technology in a controlled manner, if possible, rather than introducing it to the entire organization too fast. Of course, it is always safest to introduce the new technology on a project that is small in scope and not critical to the core of business operations, but often we do not have that luxury. Platforms are constantly changing, which adds to the challenge. An often overlooked aspect is to remember to incorporate additional time into the schedule and money into the budget for training the employees in the new technology.

CONCLUSIONS

If one is to perform rapid application development, it is even more critical to have software process improvement initiatives in place and to follow sound software engineering practices. Organizations need to employ project management fundamentals, which include the tasks of estimation, the commitment process, planning, tracking and control (measurement practices and risk management practices), software testing methodologies, and people management.

REFERENCES

- [1] Brooks, F. 1987. "No Silver Bullet - Essence and Accidents of Software Engineering", *Computer*, April, pp.10-19.
- [2] Jones, C. 1994. Assessment and Control of Software Risks, Yourdon Press.
- [3] McConnell, S. 1996. Rapid Development: Taming Wild Software Schedules, Microsoft Press.
- [4] Hambling, B. 2000. "Testing in a RAD Environment", post-conference tutorial at the Sixth International Conference on Practical Software Quality Techniques, Austin, Texas, March 2000.
- [5] Paulk, M., B. Curtis, M.B. Chrissis, and C.V. Weber. 1993. "Capability Maturity Model for Software, Version 1.1", Software Engineering Institute, CMU/SEI-93-TR-24, February.
- [6] Kasse, T. and P. McQuaid. 1998. "Entry Strategies Into the Process

Improvement Initiative”, *Software Process Improvement and Practice Journal*, Vol. 4, Issue 2, pp.73-88.

[7] Demarco, T. and T. Lister. 1999. Peopleware: Productive Projects and Teams, 2nd edition.

[8] Humphrey, W. 1996. Introduction to the Personal Software Process (SEI Series in Software Engineering), Addison-Wesley.

[9] Humphrey, W., M. Lovelace, and R. Hoppes. 1999. Introduction to the Team Software Process (SEI Series in Software Engineering), Addison-Wesley.

[10] Kasse, T. and P. McQuaid. 2000. Software Configuration Management for Project Leaders, *Software Quality Professional*, September 2000, Volume 2, Issue 4, pp. 8-19.

BIOGRAPHY

Dr. Patricia McQuaid is an Associate Professor of Management Information Systems at California Polytechnic State University in the United States. She has taught a wide range of courses in both the Colleges of Business and Engineering. She has industry experience in Information Systems Auditing in both the banking and manufacturing industries and is a Certified Information Systems Auditor (CISA). Her research interests include software quality, in particular, the areas of software quality management, software process improvement, software testing, project management, and complexity metrics. She performs consulting in these areas.

McQuaid served as the Chair for the Americas for the Second World Congress for Software Quality, held in Japan in September 2000. She will hold the same position for the Third World Congress to be held in France in 2005. She is a frequent speaker at both national and international conferences.

McQuaid has a doctorate and master’s degree in Computer Science and Engineering, an MBA, and an undergraduate degree in accounting. She is a senior member of the American Society for Quality (ASQ) and is a member of Association for Computing Machinery (ACM), IEEE, IEEE Computer Society, the International Function Point Users Group, and the Information Systems Audit and Control Association.

Evolutionary Development of Web-applications – Lessons Learned

Anette Cecilie Lien and Erik Arisholm

*Industrial Systems Development
Department of Informatics
University of Oslo
{anetteli; erika}@ifi.uio.no*

Jon Skandsen

*Genera AS
jsk@genera.no*

Abstract. This paper describes experiences from evolutionary web development projects. These experiences suggest several, simple process guidelines. The guidelines will subsequently be evaluated on new development projects, as the next step in a process improvement cycle.

We studied three web development projects out of which two included user participation.

Users, customers and developers were interviewed. The users had difficulties in making contributions before they were shown some parts of the product. Once prototypes were developed, the users continually had new suggestions and change requests during the development project. However, the change management came out of control because of lacking formal routines for handling the change requests. Furthermore, the results show that web-technology is immature. This may in turn affect the way project initiation, estimation and prototyping should be performed.

Introduction

Genera AS is a vendor of Genova, which is an advanced CASE tool for object-oriented analysis and design, dialog modelling, and automatic application generation and

database generation (Arisholm *et al.*, 1998). In conjunction with the development of the Genova *tool*, we are also developing the evolutionary Genova *process*. Such evolutionary development processes have been proposed as an efficient way to deal with risks such as new technology (e.g., web-technology) and imprecise or changing requirements (Boehm, 1988). The main idea is to resolve risks early by incrementally evolving the system towards completion instead of relying on the traditional “big-bang” waterfall approach (Royce, 1970). Thus, an important objective of evolutionary development is to identify the “real” needs of the customer as the system evolves. However, exactly *how* evolutionary development should be performed to provide the benefits hoped for remains an open research question (Arisholm, 2001). It is plausible that the process models will need a significant amount of tailoring depending on various characteristics of the development project at hand.

During the past few years, Genera AS have been involved in many web application development projects. Through the project experiences, it has become apparent that web application development may have many characteristics that require specific process support. This paper describes initial efforts to develop the *Genova Web Process*, tailored to support evolutionary (i.e., iterative and incremental) development of web applications. More specifically, experiences from three web development projects in which Genera was involved are reported. Based on these experiences, several process guidelines are proposed. We believe these guidelines may be useful for other companies in similar development projects. A simple prototype of a change management tool is also described. This development of this tool is motivated by the observed need to formalise communication paths among stakeholders in evolutionary development projects.

The remainder of this paper is organised as follows. Section 2 gives an overview of process improvement activities in Genera related to the development and empirical evaluation of the Genova process, and gives an overview of the study design. Section 3 describes experiences from the three web development projects. Section 4 recommends a set of process guidelines, based on the experiences from the analysed projects. Section 5 relates the results to existing studies. Section 6 concludes and describes future work.

Improving the Genova Process

Initially, the Genova process was defined as a scaled down version of the Rational Unified Process (RUP). We believe that such a “light-weight” evolutionary process will increase the likelihood of success in smaller development projects (Arisholm *et al.*, 1999).

At present, further development of the Genova process is underway. The study described in this paper is a part of a process improvement project funded by a Norwegian industry research project, PROFIT. The goal of a sub-project of PROFIT (in conjunction with Genera) is to develop guidelines for evolutionary development projects in general, but with emphasis on web applications development in particular. The guidelines are based on the experiences collected from three web development projects through interviews with (1) developers and project management representing the main contractor, and (2) end-users and management representing the customer. General guidelines will subsequently be incorporated in

the Genova Process. Guidelines specific to web development projects will be incorporated in the Genova Web Process. The guidelines will subsequently be evaluated, both qualitatively and quantitatively, on new web development projects. Thus, the experiences presented in this paper describes the results of the first phase of an effort seeking to improve the way web applications should be developed using an evolutionary process.

The following subsections describe the design of the experience collection phase of the Genova improvement project. The study consisted of semi-structured interviews with subjects involved in three web development projects. The following experience collection process was used:

- Project selection
- Subject selection
- Data collection
- Analysis

Project Selection

The development projects were selected based on relatively recent projects in which Genera was involved, either as a main contractor or as a service provider for the main contractor. Since the experience collection is based on the subjects' recollection of events, selecting fairly recent projects probably improves the quality of the data. Furthermore, the goal was to select quite diverse projects, in an attempt to uncover as many characteristics of web development as possible, including both positive and negative experiences. However, all of the selected projects are web-based systems that are accessible only within the customer organization's intranet. Table ALEAJS.2 gives an overview of the selected web projects. The TelMont project contributed substantially more to the process guidelines outlined in Section 4 than the other two projects. Note that the project and company names have been changed for confidentiality reasons.

Subject Selection

Several interviews were conducted with subjects selected to cover different roles in the development projects. In total, 15 persons were interviewed (Table ALEAJS.1). The subjects had central roles in the development projects, and consisted of

- developers and project management from the contractor,
- the customer project management, and
- the end-users.

Table ALEAJS.1 Overview of the interviewees

Project Name	"TelMont"	"LibTime"	"UniBase"
Project Manager	Yes	Yes	Yes
# Web Developers	2	2	1
Customer	Yes	Yes	No
# Users	4	2	None

Data Collection

Before the interview sessions, several specific questions were formulated by each of the two researchers. The questions were combined and prioritised to form an interview guide.

The interviews were recorded on a tape recorder in order to avoid loss of information. As a starting point for the data collection, the interview guide worked very well.

Other, more open-ended questions were also asked. Thus, the interviews were designed not only to elicit the information foreseen, but also unexpected types of information.

For the end-users in the TelMont project, a questionnaire consisting of 11 questions was distributed by email. Four out of the six end-users involved in the development project answered the questionnaire.

Analysis

For the largest and most complex project, the TelMont project, each question and answer from the recorded interviews were written down in detail. Although this transcription process is very time consuming, it is in our opinion essential to improve the accuracy and comprehensiveness of the analyses. It is particularly important for the unforeseen types of information, which may be scattered and intermixed within several answers to specific and open-ended questions. Each researcher used the transcribed material to perform the analysis in parallel, and wrote an analysis report based on the data from the interviews. Finally, these reports were combined into a unified analysis report. The experiences from this analysis process clearly show the benefits of performing the analyses in parallel. Each of the two researchers analysed the data using different perspectives resulting in distinct themes or categories of experiences. If only one researcher had analysed the data, important information would have been lost.

For the LibTime and UniBase projects, a less comprehensive analysis method was used.

The interviews were still recorded on tape, but no formal transcription or parallel analysis took place, because these projects were smaller and therefore simpler to analyse.

All of the analysis reports were given to the interviewees for quality assurance. The experiences described in this paper are based on the revised analysis reports.

Threats to Validity

There are a number of threats to the validity of qualitative studies such as this. The study assumes that the interviewees remember events reasonable well. Even then, different subjects may remember “facts” in different ways, amongst others because they have different experience backgrounds and played different roles in a project. Furthermore, a sufficient amount of information needs to be made available to reconstruct events and subsequently deduce lessons learned. Consequently, a certain amount of interpretation of the data is performed by the researchers. Such interpretations are biased by the researchers’ own experiences and theories. Thus, there are clearly many sources of threats to the *internal* validity of this study.

As with most case studies, *external* validity is also difficult to achieve. Ideally, the selected projects should be representative of “typical” web development projects to ensure that the lessons learned can be generalized beyond the studied projects.

To reduce the impact of such threats, detailed descriptions of the projects are

provided to enable the reader to assess whether the experiences might be of relevance for another project.

Experiences from the Web Development Projects

This section provides a detailed description of the studied development projects, outlined in Table ALEAJS.2.

TelMont

TelMont is a support system used by a telecommunications company. The support system simplifies and automates the work processes required for performing compression and optimisation activities for ISDN telephony hardware. The users of the system are professional engineers.

An external company (i.e., the “contractor”) developed the product on a per-hour-basis.

The development phase of the project required 8600 person-hours over a nine-month period. The project consisted of a total of 10 to 15 persons, in addition to some external consultants. Specialists in the customer organisation who knew the ISDN compression and optimisation work processes contributed to the analysis and design.

Table ALEAJS.2 Overview of the studied development projects

Project Name	TelMont	LibTime	UniBase
Description	Support system used by a telecommunication company. Performs compression and optimisation activities for ISDN telephony hardware.	Promotion of journals to libraries by a Norwegian Library Organization. The application supports search for magazines, and ordering and cancellations of subscriptions.	To generate economic and administrative reports from several databases in Universities and Colleges in Sweden.
#Developers	10-15, plus some external consultants	3	4
Contractor	External, paid per-hour	External, fixed price	External, paid per-hour
Duration	9 months	6 months	5 months
Effort Estimation	Exceeded effort estimates with 100%	Effort estimates accurate	Effort estimates accurate
Technology	MTS/COM, C++, Oracle database, SQLIIS, ADO, SQL	ASP/VB, Microsoft	PHP, Mimer ODBC driver, SQL
Process Model	Evolutionary – the Solution Delivery Process	Evolutionary – the Genova Process	Evolutionary, Undefined
Customer involved	Yes	Yes	Frequently asked, but

in development?		did rarely answer questions
Users involved in development?	Yes	Yes (User = customer)No

Project Activities and Milestones:

May 1999 - August 1999 (Inception)

A pilot project was initiated by the customer in May 1999. The goal of the pilot project was to determine the need for an automated software solution. The deliverable from the pilot project was an analysis report.

August 1999 - October 1999 (Elaboration)

The requirement specification activities started in August. In this phase of the project, the main work processes that the TelMont system was intended to support were discussed. The work resulted in a requirement specification and an analysis model consisting of a workflow model and some use cases, as well as a simple prototype of the user interface.

October 1999 - March 2000 (Iterative Elaboration and Construction)

In this phase of the project, a large evolutionary prototyping activity was initiated. The prototype had two purposes: 1) to evaluate the feasibility of the chosen technology platform, and 2) to elaborate requirements by providing a detailed user interface of the most important functionality. The product was modelled in UML.

The first construction increment (or prototype) in the project was finished in March 2000. This increment served as an architecture release and as a requirements specification. In October/November the contractor asked the customer to provide end-users in order to evaluate the system iteratively. However, these end-users were very important engineering resources within the customer organisation, and there was considerable debate within different groups in the customer organisation regarding how and when to assign these resources to the TelMont project. Finally, in early February, end-users were assigned to the project.

The part of the prototype intended to evaluate the architecture and technology became very large, and took considerably much more time and effort than planned. Based on the workshops with the end-users, the developers discovered that there was too much complexity in the initial requirement specification. A considerable amount of rework occurred as a result of the end-user feedback.

Both customer and end-user had access to the evolving system at all times. Thus, users could test solutions and suggest changes continuously. However, the workshops and the continuous evaluation of the system by the end-users produced a large number of change requests that were not always handled through formal channels.

March 2000 (Changed Process)

From March 2000 the evolutionary process (called the "Solution Delivery Process") was abandoned and replaced by a formal waterfall process, requiring formal acceptance of detailed specifications and code at the method level. This radical change in the project was determined necessary to get economy and time schedules under control. The contractor could no longer defend using the solution delivery process because of the

delays and the complaints voiced by the customer caused by the continuously changing requirements. The customer felt they lost control because the requirements changed continuously, without the formal documentation being changed accordingly. However, the developers strongly believe that the changes requested by the end-users were crucial in order to achieve a realistic and useful product.

May 2000

A detailed system specification was accepted.

June 2000 (Transition)

The first main release was in June 2000, and consisted of a fully operational system. At this time, the project had used 8600 person-hours, whereas the initial estimate was 4300 person-hours. The release was delayed by three months.

LibTime

LibTime is an application developed for the promotion of journals to libraries by a Norwegian library organisation. The application supports search for journals, ordering and cancellations of subscriptions. The service of ordering subscriptions has earlier been conducted manually via telephone or fax. The goal of the project was to provide a web interface as an extension of this service.

The application was developed by three developers in cooperation with two end-users.

The end-users were librarians employed in the customer organization. The developers and customer attended three meetings during the development. At the meetings the product developed so far was showed to the customer. The feedback from the end-users resulted in new change requests.

Initially, the requirements specification was made by the two end-users in cooperation with a consultant firm for several years, using an evolutionary approach.

Originally, the consultant firm was supposed to develop the whole application, but the time limits set was delayed by more than 100%, resulting in that the customer decided to change contractor. The requirements specification had then been going through several iterations, and many aspects of the upcoming product were discussed. The requirements specification had therefore been well thought through before delivered to the developers at Genera.

The product development did, probably as a result of rather mature requirements, not suffer from serious changes at the architectural level during the development at Genera. However, there were still a large number of continuous change requests from the end-users regarding details of the product. Change requests were also raised on parts of the product that already had been changed several times, because the end-users found it difficult to foresee the consequences of the change requests. The developers complained about too much time being wasted on discussions with the end-users regarding details of the change requests. The end-users did, on the other hand, find these discussions fruitful, and thought that the product would not have been adjusted to their real needs in the same way without these discussions. They are very pleased with the delivered product.

Despite the number of end-user change requests, the software cost estimate was accurate for this project.

UniBase

The UniBase system is an administrative system that has access to several economic and administrative databases in Universities and Colleges. Based on user input, the system generates reports that fit the queried data from the databases via a PHP-script. The

generated reports can be read in MS Excel or a similar application. The purpose of the project was to convert the interface of a traditional application into a web interface. The reason why the customer chose to transfer the traditional application into a web-application was to make it more accessible. Having the system in on the web implies that it is easy to distribute because the web interface eliminates the need for installing the application at every user's PC.

There was no explicit requirements specification. Instead, the contractor was given the traditional application and was asked to make the web interface as similar to this application as possible.

Four developers developed the application during a five-month period. The customer was involved in making decisions when the new solution would differ severely from the original application. In these cases the communication would at times be slow.

When the product was delivered for installation, the contractor did not hear from the customer until another two months had passed. By that time, the developers had already been allocated to other development projects, and since the contractual support period had passed, all assistance and fault correction were delayed.

The UniBase project did not follow any defined process model. This was partly due to the sparse contact with the customer, and that the "requirements specification", i.e., the traditional system that was to be transferred, was well known. Furthermore, the need for a defined process was also reduced because the project was small in size and duration. The overall cost estimates were accurate, and the product was initially delivered on the negotiated time.

Process Guidelines

This section describes recommendations based on the experiences from the three development projects. Based on these simple recommendations, we believe several of the problems experienced on the studied projects may be avoided on future evolutionary development projects of web-applications. Some readers may find the recommendations simplistic or perhaps rather obvious. Nevertheless, the actual project experiences suggest the need to reiterate the importance of recommendations such as the ones proposed in the following subsections.

Project Establishment and Planning

When estimating project costs for web development projects, be sure to account for the use of immature technology and the competency building that often is required.

- Experiences from all the projects showed that building web applications required a considerable amount of new technical knowledge. This must be accounted for when estimating project costs.
- Web development technology was experienced to be immature, time-consuming and error-prone. Consequently, it may be harder to use than development technologies for more traditional development projects.

Ensure that the anticipated functional and non-functional requirements of the application are supported by the chosen web development tools.

- Experiences from all the projects showed that, for web development tools, important functionality is less refined or absent compared with traditional (i.e., more operating-system dependent) development tools. Thus, the development tools should be assessed carefully to ensure that they enable adequate support for the anticipated functional and non-functional system requirements.

User Participation

Aim to get in contact with the end-users as soon as possible. Considerable cost savings can be achieved if the end-users are involved in the project from the inception. The end-users can contribute to and evaluate the requirements specification and eliminate unnecessary functionality before it is implemented.

- In the two projects with end-user participation, both customer and contractor found that the product had a higher quality at delivery as a direct result of the end-user participation. They developed the *right* product – a product that was applauded by the end-users.
- All of the users think that their participation was useful, and resulted in a product better adjusted to their work processes. The end-users thought that their presence in the project was necessary to ensure implementation of important requirements. The user engagement also increased their willingness to use the product after delivery, and to teach it to other end-users.
- User participation may result in more satisfied users. The end-users reported more satisfaction with the project they had been involved in, than with other similar projects without user participation. The end-users satisfaction was partly due to that they had been given the opportunity to raise their opinion, and to refuse functionality that was regarded unnecessary. They also gave important input on forgotten topics and work processes that was necessary to make a well-functional product.
- In one of the projects, much time was wasted on a too complex requirements specification. When the end-users were contacted to evaluate the requirements specification, much of the functionality was considered unnecessary and was eventually removed. The customer, developers and end-users reported that costs and time would have been reduced if the end-users had been actively consulted from the inception of the project.

- Reducing complex functionality does not only reduce costs – it also adapts the product so that it fits better to what the user really needs. In one particular case, the less complex product implied that it became easier to use *and* easier to maintain.

Do not expect instant feedback from end-users. Give them time to get familiar with the (incremental) product deliveries.

- The end-users requested changes only after some period of development, that is, when the end-users were more familiar to the product and could therefore see areas of improvement more clearly than in the beginning.

The contractor should be aware of that the end-users in most cases already have a full workload of regular work tasks. Consequently, the contractor should negotiate with the managers of the end-users to help them be relieved of some of their regular work tasks.

- In the studied projects, the participating end-users were not relieved of their regular work tasks. Consequently, the participation in the project was exhausting for the end-users. End-users from both projects reported that they had wished to be relieved of their regular work tasks one or two weeks to pay full attention to the development project. The contractor (and customer) should be aware of this, and negotiate with the managers of the end-users to help them to be relieved of some of their regular work tasks.

Prototyping

Focus early prototyping activities on technology evaluations. Clearly separate technology prototypes from the requirements prototypes.

- In one of the projects, too much effort was put into a combined technology/requirements prototype. By the time the prototype was operational, it had become very large. Because of the large amount of effort spent on the prototype, it was decided to base the actual product on the prototype. This decision were taken because it was economically impossible to defend why the chosen technology should be replaced, even though the prototype proved that the technology (Microsoft/Oracle) had severe difficulties around integration issues. The technology was also shown to be error-prone with obvious limitations. We therefore recommend making a simple technology prototype during the inception of the project to test the chosen technology, and separate this prototype from other kinds of later prototyping activities.

End-users are potentially highly skilled evaluators of product functionality. However, the end-users may not be technically skilled, and consequently may have some difficulty understanding technical details before they can be demonstrated in practical terms. This gap in skills between developers and end-users may limit communication efficiency. Use prototyping as a way to bridge the gap in skill level between the end-users and the developers.

- The users in two projects reported problems with foreseeing the consequences of their choices regarding technical details. They could thereby change their mind when the change request was implemented and the results were reviewed. This was

annoying and confusing for the developers. Developers should be aware of this, and for complex change requests it may be advisable to make a tiny prototype and let the end-user evaluate it before putting too much effort into the implementation of the change request.

- The users in two of the projects described it as difficult to imagine the resulting product when they reviewed the requirements specification. It was easier to evaluate the requirements specification as the development had been going on for some time, and they saw some parts of the coming product. It was first after some time they could envision the final product, and thereby got the courage to suggest changes.
- A project leader for one of the customers describes the prototypes as an essential tool for recognising requirements and visualising the upcoming product to the end-users. Without the end-users' review of the prototypes, much unnecessary functionality would have been implemented.

Change Management

Better communication regarding software change management may increase mutual agreement around revised costs and time schedules. We recommend that a formal change management process is agreed upon and followed by the customer and the contractor. To ensure process conformance, consider using a change management tool to support the communication between end-users, customer and contractor. Appendix A describes details of such a process model and a supporting tool.

- In evolutionary development, problems with cost control as a result of change requests from the end-users may arise. While new changes are suggested by the end-users, thereby adjusting the product to better fit their needs, the customer may not see these changes as merely advantageous as the costs and time schedules may get out of control.
- Two projects experienced severe communication problems and considerable frustration as a result of not agreeing on a formal change management process. All project members reported a need for some kind of formalized process to keep track of the changes. The developers in one project were confused because the customer and end-users regularly changed their mind, and many parts of the product had to be changed several times. In the second project, the customer was very displeased about not being consulted about whether certain change requests were acceptable from an economical point of view. Both problems may have been avoided had a more structured change management process been followed. The process should formalize communication paths between parties (contractor project management, developers, customer, and end-user) to get change requests and costs under control.
- In one of the per-hour-based projects, the customer was very frustrated with the frequent changes to the requirements specification, because they caused project delays and extra costs. The continuously changing requirements specification was a source of discussion. The project manager at the customer felt that he lost control because the requirements changed continuously. However, the developers strongly believed that the changes requested by the end-users were necessary in order to adjust the product to their end-users needs. The result in the end was a cost discussion. The discussion did not get a closure until the contractor discarded several hours of work. A more formal change management process may have alleviated some of the

experienced problems.

Related Work

The results presented in this paper mostly confirm the results from related empirical studies, although there are some differences as well. The study conducted by (Emam *et al.*, 1996) suggests that user participation is advantageous in system development with uncertain requirements. When uncertainty is low, the beneficial effects of user participation diminish. On such projects, user participation could instead be annoying to the end users because they feel their efforts are redundant. This resentment may bring out *reductions* in quality of service as user participation increases. These results are in accordance with our observations of the successful user participation in TelMont and LibTime versus the non-participative approach of the UniBase system. In the UniBase system, the end users were unwilling to contribute to the development. The requirements specification was from their view given with no uncertainties, that is, they wanted their traditional system on web with no changes in functionality or design. They seemed very satisfied with the final product despite their sparse contact with the developers through the development process. In the two other projects, however, the users did not know exactly what requirements and functionality the final product should include. The requirements specifications were refined and changed several times by prototyping their suggestions and through dialogues with the developers. The end users found that their participation in the system development were necessary and useful in developing a *right* product that matched their needs, because they were unable to see what they really wanted before parts of the product were prototyped. Their presence also reduced the developers' uncertainty regarding the requirements specification.

The study conducted by (Zamperoni *et al.*, 1995) indicates that sometimes users or customers had a specific vision of the future system, but are unable to formulate this perception properly to the developers. This problem could be improved by using evolving prototypes. The prototypes could be useful in communication – in being a mediator to reach agreement with the users and their expectations. In accordance with our results, Zamperoni *et al.* also found that the product is more likely to be accepted among the users if they are involved in the development process. However, they stress that short communication channels between users and the development team are important prerequisites to increase the likelihood of acceptance. Similar results have also been reported by (Ehn, 1993). He argues that the end users are highly qualified in evaluating the product functionality, but that they do not necessarily have the technical skills to understand the requirements before it is demonstrated in practical use. Ehn points out that this gap in technical skills may be bridged by the use of prototypes. We made the same observation in our study, as the users pointed out that it was easier to contribute after a while, when they were able to see some parts of the final product, not just a technical specification.

The study in (Lichter *et al.*, 1994) presents a critical view of end user involvement. They found that developers may do the mistake to encourage the end users to voice all ideas and wishes that come into their mind when they evaluate a prototype. This may lead to the incorporation of every conceivable function or design option, and thus increases the complexity and reduce the usability of the final system. This

was *not* the case in our study. Especially in the TelMont project, we rather found that the user participation reduced the complexity of the requirements specification and the final product.

On the TelMont project, user participation resulted in a less complicated requirements specification and made the product both more maintainable and easier to use. However, as Bratteteig (Bratteteig, 1997) suggests, reducing functionality in this way may not be an easy decision to make for the developers. The end users' needs of having the easiest possible product to use may collide with the developers' wishes of making a product that incorporates new technology and impressing functionality that is more challenging for them to code. To the end users, technical elegance may not be as important as it is to the developers. On the TelMont project, this tension was experienced, but was solved smoothly. The management wanted a product that incorporated many technical details and several ways of navigating. The end users, on the other hand, wanted the system as simple as possible, with one restricted way to navigate through the product. It is a challenge to the developers to accept this, but in TelMont they chose to go for the end users solution, and the result was highly satisfied users.

Hohmann (Hohmann, 1997) argues that the design of the developer organization's processes is reflected in the design of the products they make. He reasons that if the developer organisation has well organised communication channels and is properly structured, it will be reflected in the design of their products. A well-defined organisational structure constitutes a shared belief amongst the developers of how the job should be performed, and this may in turn lead to that the product is well structured. A parallel can be drawn to formal change management, e.g., as proposed in this paper. A good structure and well-defined communication paths for the change requests may structure the change management, and avoid that it comes out of control.

Forsyth (Forsyth, 1998) shows by empirical material that planning a strategy to use in development of products (e.g., process improvement) contributes to better verbal communication, increased motivation and higher flexibility in conducting the planned tasks. If we draw the parallel to change management, planning the change management process may be advantageous to both communication around – and structuring of – the change requests so that they do not run out of control.

Conclusions and Future Work

This paper described current efforts to develop the Genova Web Process, tailored to support evolutionary development of web applications. Based on experiences collected from three web development projects, specific process guidelines on important topics such as initiation, planning, user participation, prototyping and change management were proposed. We believe these guidelines may be useful for other, similar types of projects.

Future Work

One of the guidelines suggests a need for a better way of handling change requests in evolutionary development of web applications. Uncontrolled changes had been a source

of frustration in the projects. As a response to these observations, a simple but formal change management process is under development. A tool (the "change management prototype") that supports this process is being implemented. Some details of the process and supporting tool are described in Appendix A.

At present, the change management prototype will be evaluated on a new web development project in which Genera is the contractor. During the project, the prototype will be used by the customer, end-users and developers. Interviews will be conducted to evaluate the effect of the formalised change process tool, and to determine whether it should be refined and developed further into an integrated, commercial part of the Genova Web Process.

Acknowledgements

We thank the interviewed participants of the three development projects for spending valuable time and for allowing us access to potentially sensitive information. The research project is financed by The Research Council of Norway through the industry-project PROFIT.

References

- Arisholm, E. (2001). Empirical Assessment of Changeability in Object-Oriented Software. PhD Thesis, University of Oslo.
- Arisholm, E., Benestad, H.C., Skandsen, J. and Fredhall, H. (1998). Incorporating Rapid User Interface Prototyping in Object-Oriented Analysis and Design with Genova. In: Proceedings of NWPER'98 Nordic Workshop on Programming Environment Research, Sweden, pp. 155–161.
- Arisholm, E., Skandsen, J., Saggi, K. and Sjøberg, D.I.K. (1999). Improving an Evolutionary Development Process - A Case Study. In: Proceedings of the EuroSPI'99 (European Software Process Improvement Conference), Pori, Finland, pp. 9.40–9.50.
- Boehm, B.W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21 (5), 61–72.
- Bratteteig, T. (1997). Mutual Learning. Enabling Cooperation in Systems Design. In: IRIS'20.
- Ehn, P. (1993). “Ch.4: Chandinavian Design: On Participation and Skill”. In: *Participatory Design: Principles and Practice*. Schuler, D.N., Aki (editors), Lawrence Erlbaum, pp. 41–77.
- Emam, K.L., Quintin, S. and Madhavji, N.Z. (1996). User Participation in the Requirements Engineering Process: An Empirical Study. *Requirements Engineering*, 1996 (1), 4–26.
- Forsyth, D.R. (1998). *Group Dynamics*. Brooks/Cole Publishing Company.
- Hohmann, L. (1997). *Journey of the Software Professional. A Sociology of Software Development*. Prentice Hall.
- Lichter, H., Schneider-Hufschmidt, M. and Zullighoven, H. (1994). Prototyping in Industrial Software Projects - Bridging the Gap between Theory and Practice. *IEEE Transactions on Software Engineering*, 20 (11), 825–832.
- Royce, W. (1970). Managing the development of large software systems: Concepts and techniques. In: Proceedings of IEEE WESTCON, Los Angeles, pp. 1–9.
- Zamperoni, A., Gerritsen, B. and Bril, B. (1995). Evolutionary Software Development: An experience Report on Technical and Strategic Requirements. Technical Report TR-95-25, Leiden University, The Netherlands.

Appendix A: The Change Management Tool

This appendix describes a simple, but formal process to manage the communication paths and activities for handling changes in an evolutionary development process such as Genova. Figure ALEAJS.1 depicts an example workflow handling end-user change requests.

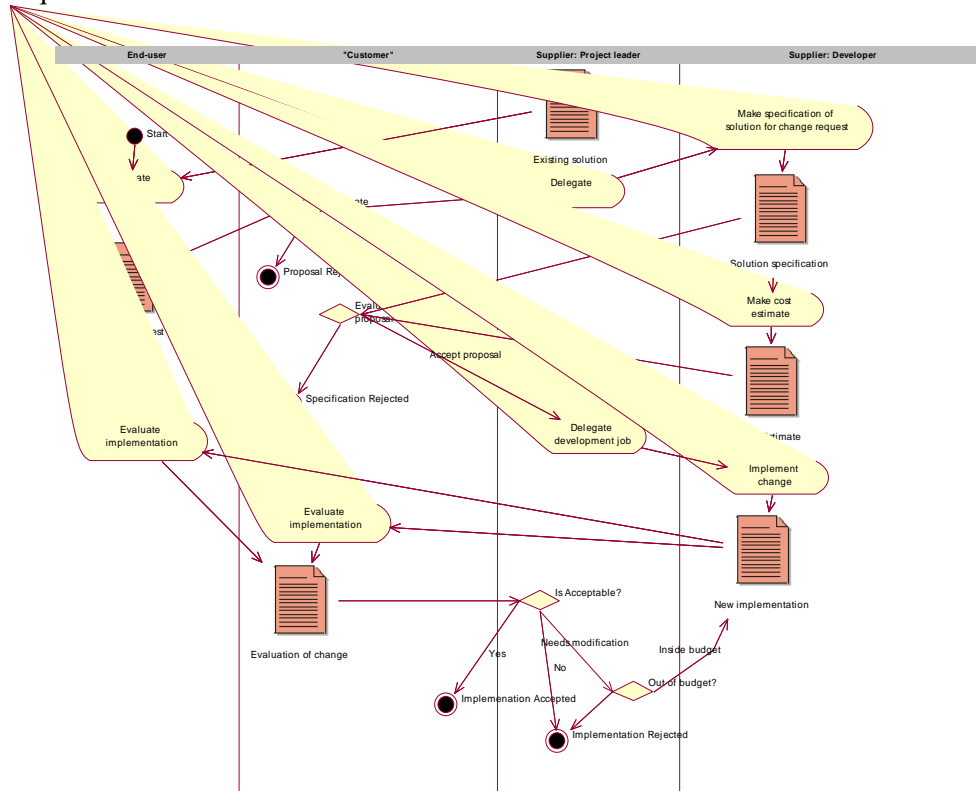


Figure ALEAJS. 1 Example workflow for change requests submitted by end-user. The workflow describes activities, deliverables and roles using a UML activity diagram as the notation.

The process is supported by a simple change management tool that is situated on the web (Figures ALEAJS.2 and ALEAJS.3). Change requests can be described and submitted (e.g., from end-users) into a shared change management database. The customer can retrieve new change requests regularly from the database, and approve, postpone or reject the change requests for further investigation. The project manager in the contractor organisation can retrieve change requests approved for investigation from the database, specify a solution, and send the change proposal back to the customer for evaluation. If the change proposal is accepted by the customer, the new change task is delegated by the project manager of the contractor to a developer for implementation. When the developers have been assigned to a specific change task, they can look up the corresponding change request from the database, and contact the change requestor (e.g., end-user) to get a further explanation of details. The developers also log their effort spent on the task. These data may be shown to the customer to explain revised estimates.

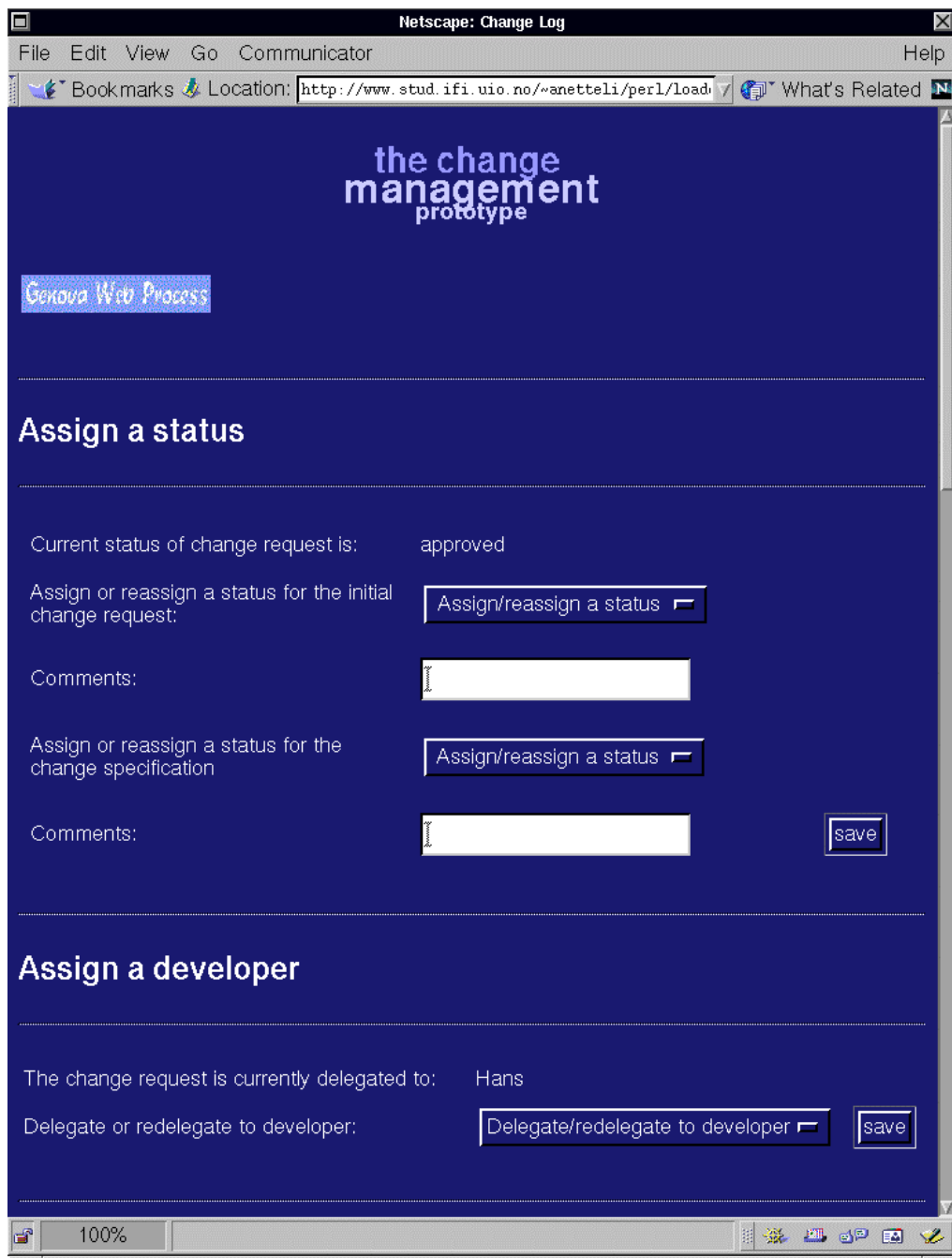


Figure ALEAJS. 2 Tool for change request status assignment and delegation

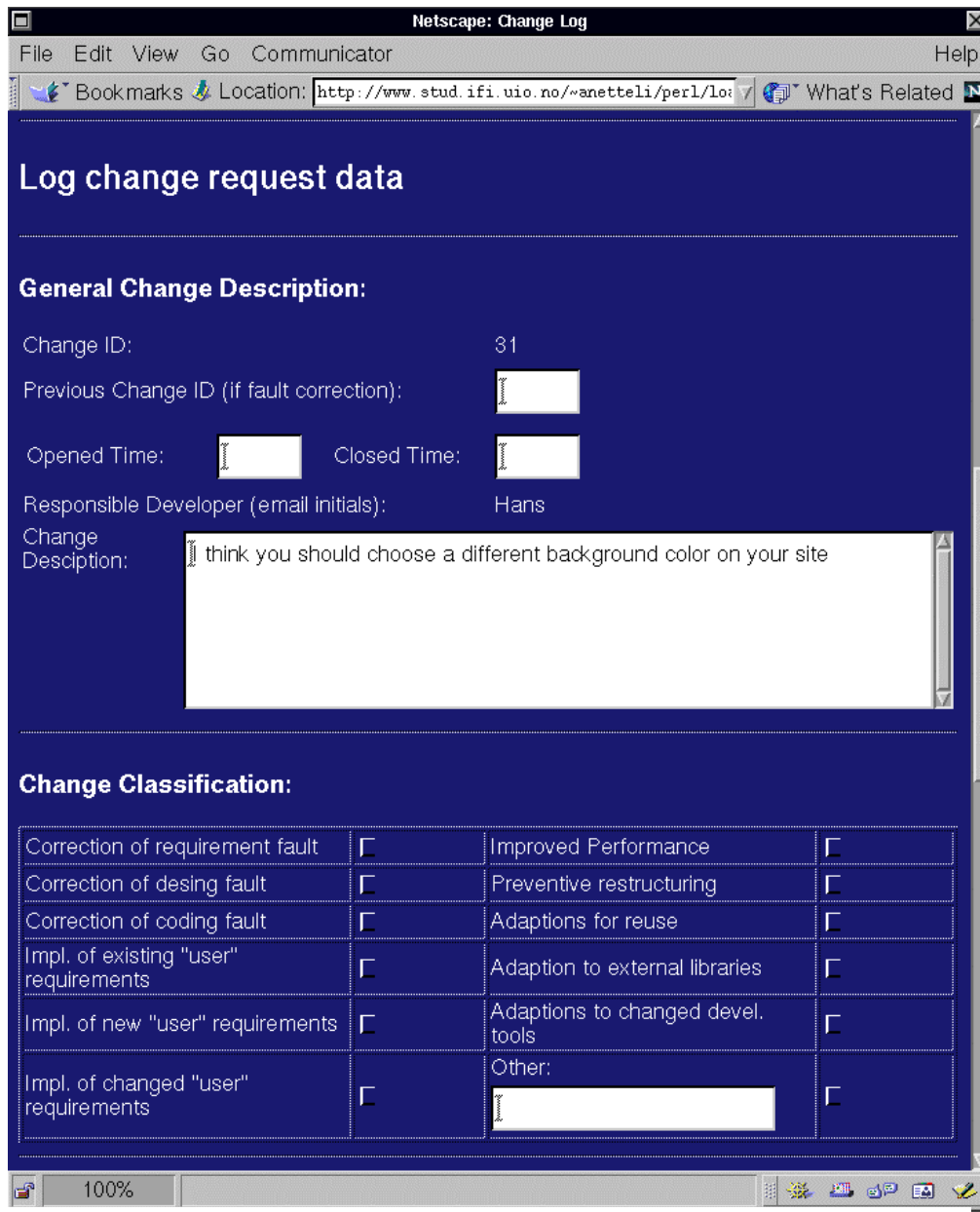


Figure ALEAJS. 3 Parts of the developer's log

Session 5 - SPI and Analysis

**Session Chair:
Mads Christiansen, Delta,
Denmark**

Post Mortem - An Assessment of Two Approaches	5-2
Integrating SPI and Assessment Models to Facilitate Measurement and Understanding	5-20

Post Mortem - An Assessment of Two Approaches

Tor Stålhane
Torgeir Dingsøy
NTNU, Norway

Geir Kjetil Hanssen
Nils Brede Moe
SINTEF, Norway

Abstract

Learning from experience is the key to successes for all that develop software. Both the successes and the failures in software projects can help us to improve. Here we discuss two versions of Post Mortem Analysis (PMA) as methods for harvesting experience from completed software projects, which can be part of a larger knowledge management program. The two methods are tailored for use in small and medium size companies and are conceptually easy to apply. In addition, they require few resources compared to other methods in the field. We think that the methods are useful for companies when they need to document their knowledge, find improvement actions and as a start of systematic knowledge harvesting.

Introduction

An obvious way to improve a software development process is to learn from past mistakes. In practice this has turned out to be easier said than done, but the method of post mortem analysis - also called post mortem reviews or PMA - is one way to achieve it. There are several companies already doing this, such as Apple Computers [1], Rolls Royce [2] and Microsoft [3]. What they all have in common, however, is a rather elaborate and costly process.

We are mostly dealing with small companies, with small projects. The overhead from an approach as ambitious as the one used by, for instance, Apple Computers would be way beyond the reach of most of our customers. We needed something simpler and wanted to try out the two rather simple approaches described in this paper. In order to see if they were useful we tried them out in two companies. Our goals were to see if:

- The approaches worked - i.e. if anything useful came out of them.
- They worked in different ways - i.e. were the information that came out of them different in amount, scope or usefulness.

The research was conducted in close collaboration with industry. We collected experience from real software projects in a real environment, over which we had only limited control. What we present here is thus not a scientific evaluation but our experience from practical improvement work. Thus, what comes out is not hard science in the strict sense but practical experience for practical people working in the area of software process improvement.

The rest of this paper is organised as follows: First we describe our approaches to PMA. Then we provide a short description of the two companies where we tested our approaches, and what we found out when we did this. Finally, we discuss the usefulness of the two PMA methods and the usefulness of what came out of them.

Harvesting experience

Learning from experience

To improve software development, it is important to help software developers to learn both from positive and negative experience, i.e. to become what is often termed a “learning organisation” [4]. People usually learn best when they can relate what they learn to their ordinary work. Thus, experience collected from a well-known environment will have a greater learning effect than experience collected from unfamiliar environments.

Learning from your own experience requires that you be allowed to reason about tasks that have been completed to see what went well and what did not, and also why things happened the way they did. Thus, it requires observation and reflection [5].

To cover more material, we can learn from other people’s experience as well as our own. Thus, it is a goal to transfer experience from one project to another in order to make individual and project-based learning helpful for, say, a whole organisation. An important part of the problem is how this experience can be collected effectively. We will now turn our attention to this question.

The post mortem analysis - PMA

As a concept, the PMA is simple - gather all participants from a project that is ongoing or just finished and ask them to identify which aspects of the project worked well and should be repeated, which worked badly and should be avoided, and what was merely “OK” but leave room for improvement. In addition, we need techniques that can be used to elicit experience and document it in such a way that they can be made verifiable and reusable in the most effective way. Reuse of

experience is, after all, one of the reasons for doing a PMA.

A PMA can be done in many different ways - varying both in goals and degree of formality. A PMA can be focused on harvesting experience that is available from a single activity or process step or it can try to catch all experience available from a project. Data collection can be done as semi-structured interviews, or as a multi-step defined process, for instance consisting of a group process supported by affinity diagrams, followed by a root cause analysis (RCA), see [6], using an Ishikawa diagram.

Whatever form the PMA might take, one point is important - the consideration of the influence of the project's environment. No experience is reusable without considering these factors. It is often the case that an action undertaken in a project, which led to unfavourable results, was not bad in itself - it was just not a smart thing to do in the current environment.

Given its loose form and lack of formalities, many people find it almost wrong to dignify PMA with the word "analysis", which they think should be reserved for something more scientific. The loose form and simple concept is, however, a strength and not a weakness. It gives us a method that is flexible, easy to learn and apply and containing a minimum of formalities to take care of. In this way everybody in the organisation can participate and not just as information providers. This is crucial in order to support the TQM philosophy - that quality and quality improvement is everybody's responsibility; it is not the domain of the QA department alone.

Thus, we have decided to use just a few simple techniques. One or more of these techniques have been used for all the PMAs that we report from in this paper:

- KJ, which is a structured brainstorming technique [6].
- The Ishikawa diagram – also known as the fishbone diagram [6].
- Structured interviews, where the structuring factors are:
 - The purpose of the PMA – for instance causes for underestimation of costs
 - The structure of the project – what did we do?
 - The structure of the process – how did we do it?

Spacetec	InfoStream
All project members participated	Only those who did the estimation participated
Interview with the PM to get background information (1)	Study project documents to get background information (a)
Introduce the concept of PMA to the participants (2, b)	
Perform a KJ (3)	Perform structured interviews of project participants (c)
Prioritise the items from the KJ (4)	
Prepare the KJ results for an RCA (5)	-
Perform an RCA by using the Ishikawa diagram (6)	Register cause-effect connections during the interviews (d)
Write final report (7, e)	
Present final report and get feedback from participants (8, f)	

Table 1 : The PMA processes used in the two companies.

The process used in the two projects differed, partly due to company constraints, partly due to each researcher's preferences. The way the PMAs were performed is summed up in table 1 above.

Presenting our results

When presenting our results we have followed the same format for both companies. We start with a short description of the company before describing how we performed the PMA and present some typical results. In addition to the PMAs we had some activities - called support activities -, which improve on the results from the PMAs by supplying some extra facts. In this way, we are not depending only on the participants' memory. For Spacetec, we did the interviews before the PMA, while for InfoStream a GQM planning session plus data collection and analysis were done after the PMA. In fact, some of it is still going on.

In order to present the results from the two companies in a uniform way, we have chosen to present the supporting activities after the examples for both companies. In this way, we keep a better focus on the main issues of this presentation, which are the PMAs.

PMA at Spacetec

Description

Spacetec AS is one of the leading producers of receiving stations for data from meteorological and earth observation satellites. They have developed a considerable expertise in delivering turnkey ground station systems, consultancy, feasibility studies, system engineering, training and support.

Spacetec has as an overall goal to increase knowledge transfer between projects, and has chosen estimation of software project costs as its first focus area. The company will build a knowledge repository with cost information on previous projects. One of the goals of the internal improvement project at Spacetec is to improve the estimation accuracy of technical work packages so that the deviation between real and estimated cost will be within 20%.

What we did

Two PMAs were carried out for two typical projects. The first projects had lasted for almost a year, and was 75% finished when we did the analysis. The second project was finished less than a year ago. Both projects built software for analysis of image data from satellites. The PMAs were performed as described in the Spacetec-column in table 1.

Each PMA lasted approximately four hours and we had a feedback session for one hour the following day. Steps 2 and 4 lasted only for one hour each since they were highly creative. In step 3 – see table 1 – we gave each participant a number of post-it notes and told them to write down an item that was either a problem or a success on each post-it note. We then gathered in a meeting room and asked each participant to attach each post-it note on the whiteboard. In addition, they should explain to the other participants what the issue was and why it was important. Post-it notes that were related should be placed close to each other. We used a tape recorder during steps 2 - 4, and the results were transcribed and later inserted into the report. Giving the participants quick feedback was of great value both for the participants and for the researcher. The transcribed material, in particular, created important discussions.

We wrote a post mortem report on the project, containing an introduction which described the process, a short description of the project that we analysed, how the analysis was carried out, and the results of the analysis. The result was a prioritised list of problems and successes in the project. We used statements from the meeting to present what was said about the issues with highest priority, together with an Ishikawa diagram to show their root causes. In an appendix, we included everything that was written down on post-it notes during the KJ session, and a transcription of the presentation of the issues that were used on the post-it notes. In total, this report was about 15 pages long.

The total amount of resources needed for a PMA were three persons selected from the project participants. The total work needed was 35 person hours, distributed as follows:

- 4 person hours for the researchers' preparation.
- 20 person hours for the PMA session - researchers and project personnel.
- 6 person hours for writing the report from the interviews - researchers only.
- 5 person hours for feedback and updating - researchers and project personnel.

Some examples

One result from one of the KJ sessions was four post-it notes grouped together and named “preparatory work.” They are shown in the lower left corner of the results from the KJ process – see figure 1. The arrows indicate relationships between the classes and tell which success factors that influence other success factors.

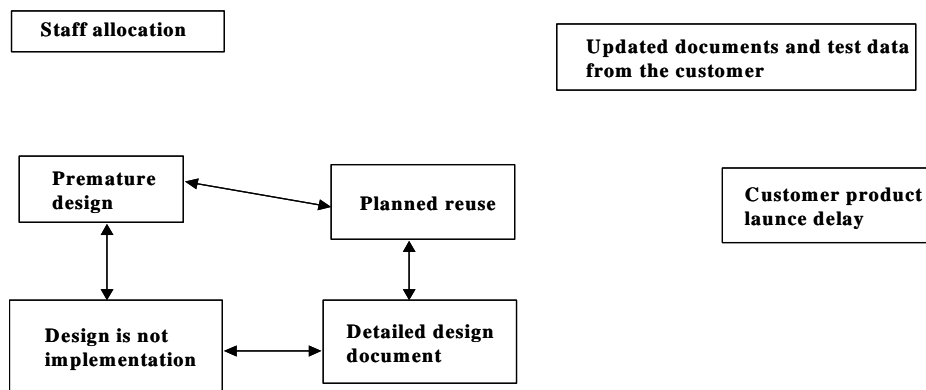


Fig. 1: Post-it notes showing problems in a project.

During the PMA meeting, the developers explained the problems related to “preparatory work” in the following way:

- *Design before “full insight”. I dare say this as well: It is OK that we are supposed to reuse, but if we had known – now I was not participating in the design, so it is wrong of me to say it - but if we had full control we could have made an architecture that was in fact possible to use later on. (“Premature design”)*
- *We put down a lot of effort into the design of the processor [a software module for analysing satellite data], and then it turned out when we were to implement it that we did not at all use the design that we had planned of in the document, so it was maybe a bit stupid to use a lot of time to... and then we don’t use it afterwards. (“Design is not implementation”)*

When we later tried to find the root causes for the “preparatory work” problem, we ended up with the following Ishikawa diagram:

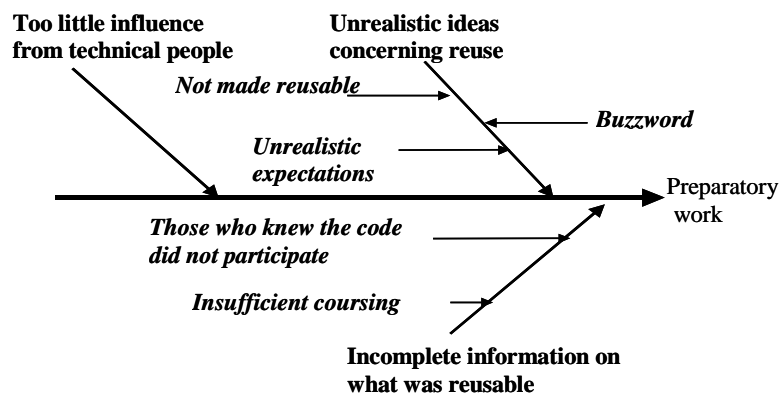


Fig. 2: Ishikawa diagram for “Preparatory work”.

The root causes for this problem, as the developers saw it, was that people in the company in general had an idealised conception of reuse. Other root causes were that the company did not have a good enough overview of what was reusable from

earlier projects and that the technical personnel did not have influence on the project activities where issues on reuse were discussed.

We then asked people to come up with suggestions for how to reduce this problem:

- Get an overview of reusable components in the company.
- Teach people higher up in the organisation about reusability.
- Update or write a new detailed design document for this project.

These suggestions on short- and long-term improvement actions were documented in the report. At the end of the meeting, we asked people how they felt about the results, and got feedback like: *The issues about contracts with the customer – I think we are too kind. There is so much that is mentioned in the contract that... it is not supposed to be like that.* In general, people said they were made aware of new issues, and were able to see smaller issues more in a larger context.

The next day, we went through the report from the meeting together with the people who attended, as well as the project manager. Some of the participants disliked that we gave a complete transcript of part of the meeting, but after some discussion, they agreed that this is something that could trigger new discussion on relevant topics. We got positive feedback on the way we organised the review, which differed from the project completion reports that the project manager would normally write alone:

One of the problems with the project completion report is that the project manager sits down in the end and sum up what went well and what went badly. And it is one man who remembers what [...] has happened the last two years. Then you have lost a lot. You do not collect a lot of experience. It is the large problems that are already discussed – things that have gone really badly. But if you do this kind of an analysis several times during the project, then we agree – it shows how... you have to think things through!

Support activities

As a basis for the PMA analysis we first had interviews with another eight persons from the company (developers, project managers, management and marketing personnel). Focus in these interviews was the estimation process and how the projects were accomplished compared to the initial plans.

We also conducted an analysis of the project completion report from 12 projects. This resulted in the following improvement suggestions:

- Introduce a 28% contingency budget on each work package to decrease the probability of cost over-runs.
- Do a focused PMA to examine why the work packages "management" and "QA" were strongly underestimated in one project, and why "coding" and "unit testing" are strongly underestimated in general.
- Do PMAs on all future projects in order to increase learning in general.
- Use a standardized structure of work packages to make projects more comparable – thus improving the possibility for comparative data analysis.

PMA at InfoStream

Description

InfoStream is a medium sized Internet consulting company that recently became part of the international Integra group. Most of the customers are in the financial, energy, media, telecom, and manufacturing sectors. InfoStream has no product line, but develops tailor made solutions for their customers.

Many of the projects of the recent past have missed on the estimates. There have been several delays in delivery and a tendency to overuse resources in the projects. This has serious economic consequences since the estimates are used to set a fixed price. Estimation precision is considered to be an important success factor, and the company is investigating their projects to find what causes these problems.

What we did

In order to perform PMAs in this company, we asked our contact person to select three typical projects that had been finished less than a year ago. This limit was imposed in order to make sure that the project experience is still available with a reasonable certainty. The three PMAs were focused on estimation and were performed as described in the InfoStream-column in table 1.

Each project PMA lasted approximately five hours, including a lunch break in the middle. During the interviews and discussions, we used a whiteboard and a flip-over to document the points as they surfaced. The flip-over was used as a collective short-term memory for the group. Except for the structure imposed by the project's work breakdown structure and the focus provided by the goal of understanding the reasons for incorrect estimates, no further structure was imposed. Thus, the notes from each PMA differs somewhat when it comes to internal structure. For instance: for one project, the participants identified the most important cause for cost overruns for each activity, while this was not done for the other two.

When the interview was finished, the researchers took all the material back to their office and structured and wrote down all the points raised. The points were numbered so that cause-and-effect relationships could be documented. This was done by inserting statements like "The estimate was increased by 40 person hours. This was caused by point 53". See also the chapter "Some Examples" below. The resulting report was sent to the participants for feedback.

The researchers went through all the three PMA summaries and extracted all the registered information that was related to the PMA focus - improving the estimation process. Afterwards we defined a slogan for each information point and wrote them down on post-it notes. This was used as an input to a KJ process undertaken by the two researchers in order to analyse the information. This way to use KJ is in line with the original way to use this method, see [7].

Figure 3 shows all the results from this KJ process. Each box contains several experiences but these are not shown in order to keep the diagram simple. The categories contained in each box are supposed to be the key success factors, and we believe that if the company paid enough attention to them we would not have experienced the poor quality of the estimates that we actually did.

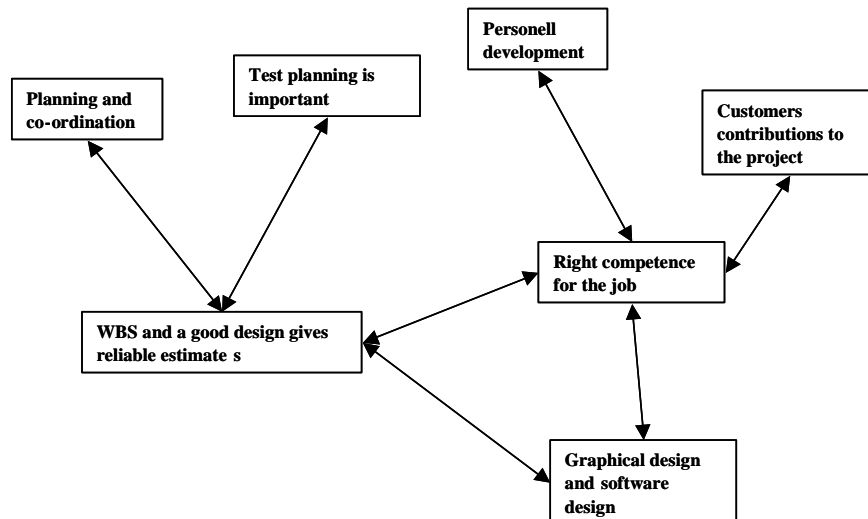


Fig. 3: Post-it notes showing the summary of problems related to estimation

The total effort for each PMA was 45 person hours or approximately one person-week per PMA. This should not frighten anyone if the work results in a substantial improvement in the way the company do their project estimates. The costs were distributed as follows:

- 5 person hours for the researchers' preparation.
- 20 person hours for the interviews - researchers and project personnel.
- 12 person hours for writing the report from the interviews and performing the final KJ - researchers only.
- 8 person hours for commenting and correcting the report - researchers and project personnel.

Some examples

The following are examples of the notes taken during two of the PMAs. All items are numbered so that it was possible to cross-reference them and thus to construct cause-and-effect chains.

PMA 1

1. The developers controlled the concept development themselves (because of 2).
2. The ideas for the product were too abstract for the customer (caused 1 and 4).
3. The developers got little feedback from the customers (because of 2).

4. The customers felt that they could change the concept later (caused 3).
5. The customer representatives were few and positive. These were the right people (caused 6 and 7).
6. The customer's participation was controlled by a subsidiary of InfoStream (because of 5).
7. The developers needed few meetings with the customers (because of 5).

PMA 2.

15. Too little time was allocated to testing.
16. InfoStream was under heavy pressure from their customer concerning time of delivery (caused 15. see also 10).
17. The customer was supposed to take full responsibility for the system's test.
18. More errors than expected – confirmed by the test reports.
19. InfoStream has later tried to improve their test planning for this customer (see 17).

Already during the first reading of the PMA reports, the participants were able to identify several improvement possibilities. The following is just a small sample:

- The developers must interact with just a few customer representatives - preferably only one. Otherwise, they will receive a large amount of uncoordinated, often conflicting comments on details in the system under development. To handle this consumes considerable - mostly unproductive - resources.
- Once set, it is next too impossible to change the customer's expectations. Thus, we must be careful when suggesting solutions.
- The graphical designers must co-operate with one or more software developers during the design of all web pages.
- The number of correction / feed back meetings with the customer must be fixed and stated before the estimates are made.

Support activities

Experience and experiments have shown that most persons have a tendency to be selective in what they remember. The reasons for this are legion and will not be discussed here. An interesting, albeit provocative, discussion on this topic can be found in [8]. In order to have a broader basis for our final conclusion and recommendations, we decided early in the process to collect measurement data for a project that was about to start at that point in time. We ran a GQM (Goal Question Metrics) process [9] in order to develop a measurement plan. Another reason for combining PMA with GQM is that from a data analysis point of view, PMA is mainly a bottom up technique while GQM is a technique that is mainly top-down. Thus, one should expect the two methods to give a more useful and complete picture of the estimation process than either one of them would achieve when applied alone. We discovered later that others have also arrived at this conclusion, albeit via another route - see [10].

Discussions

PMA experience

Our most important experience from doing PMAs is that the participants later told us that it was a highly positive experience. Since the concept of PMA is easy to grasp, everybody could participate. In addition, the developers all felt that they got something back – better insight, ways to improve their job and so on.

It is our experience that PMA is a method that is easy to use. There are no complex routines or tasks that have to be explained in advance, the method relies on the participant's intuitive understanding. A PMA will, however, create a large amount of information. Some of it is important and some is not. The job of separating the two can be both time consuming and challenging.

There are some ground rules that should be applied in order to get a successful PMA:

- The process needs to be structured – it is not a free-for-all happening. On the other hand, too much structuring – for instance extensive use of time boxing – was considered to be negative.
- The PMA will work better with an external facilitator since internal personnel may hesitate to bring up sensitive issues.
- If we are doing a general, catch-all PMA, people from all parts of the project need to participate. It is for instance not enough to have participants from those who did the coding if someone else did the design.
- If we are doing a focused PMA, the process needs to be steered – sometimes quite strongly - in order not to go astray and wind up as a general PMA or general lamenting over everything that is bad in the company.

The first decision needed if we want to do a PMA is whether it should be focused or not. Both focused and unfocused PMAs have their strong and weak points that should be considered before we make our choice.

- An unfocused PMA will usually give more surprises since it covers a broader area of experience. Thus, it is always a good idea to do a PMA as a two-stage process - first a general PMA in order to identify all the important issues and then a focused PMA for each issue afterwards.
- A focused PMA is best if we want to understand or improve a single activity. It should be done with just a few participants since the discussion must be steered in order to keep the focus. Few participants will favour structured interviews combined with discussion sessions. A focused PMA will in general require that the facilitator have in-depth knowledge of the field under discussion. This is consistent with the PROFES experience – see [11].

We started out by believing that PMAs would work best in a project-oriented organisation without a rigid control structure or an excessive hierarchy. We later learned, however, that the US Army – and maybe other military organisations – uses their own version of the PMA – called After Action Review (AAR) – with great success [12]. These authors have a more narrow definition of a PMA than we do. In their opinion the AAR is not a PMA since it is done on a regular bases during the activity's lifetime. Anyhow, the important point is that it seems that the

major factor is whether the organisation want to improve itself by learning from past mistakes, not how it is organised.

For small companies - less than five developers, say - it might be beneficial to let all employees participate actively in the PMA whether they were involved in the project or not. This will make everybody aware of quality problems and the ways to solve them. It will thus strengthen the understanding that quality is everybody's business.

At least two problems are still open:

- The PMA way of learning from experience might not work well when we have several cultures present in the group. A case in point is a PMA where both software developers and graphical designers participate. On the other hand, divergent groups can also be the basis for constructive discussions.
- How can we perform PMAs when the participants are geographically distributed? This can be a problem today and will be even more so as we get more and more distributed or virtual organisations.

What did we learn

What has been the benefit for the two companies? What have they learned and what will they do different the next time? The feedback after the sessions tells us that the participants felt that they got useful information through the PMA sessions and that they discovered new aspects, or at least was able to express knowledge that earlier had only been tacit.

- *Documenting knowledge* - Documenting a company's knowledge is always a difficult task. By using PMA it became much easier for the companies to document knowledge. The resulting documentation also gave valuable insight into the development process. The knowledge harvested in the process came from several persons in the projects. This made the conclusions more deeply rooted in the organisation than for example an experience report.
- *Source for improvement actions* - The results of the PMA sessions lead to several improvement actions. In Spacetec for example, after discovering in a PMA session that the customer delayed the project, and did not provide test data according to the contract, they sent a Contract Change Note to the customer regarding the amount of hours to be used in the project. Since this method gives fast feedback it is possible to get some "quick wins". You can have a PMA meeting one day and a list of improvement possibilities the following day.
- *A system for harvesting experience* - PMA helped the companies in the process of developing their own system for harvesting experience. Because of the strong involvement from the developers, there was an increasing focus on "learning from experience" in the companies. The motivating factor of the PMA also makes it easier for the companies to implement a system for knowledge management. PMA will be one of the central techniques in such a system.

What would we do different the next time?

- *Preparation Work* – If using interviews, it is important to have good knowledge of

the project in advance, in order to ask specific and relevant questions. In the KJ phase of the Post Mortem, it is important to be open and leave the word to the participants. Here, being able to moderate discussions is the main virtue.

- *PMA Process* – How the PMA is carried out depends on the number of participants, the complexity of the issues under investigation, and what the focus of the PMA is. Using the KJ method requires at least three participants in order to have enough material to discuss. If the issue under investigation is complex, it might be better to use interviews because it is more difficult to keep people on track using an open process such as the KJ. To get everyone to participate, the KJ method is working very well.
- *Participant Activation* – Some of the developers did not participate much in the RCA phase. It could help people to be more active if one of the participants were moderating this session. Another possibility is to have participants working in groups first, and then present results in a plenary session afterwards.
- *Documentation Techniques* – The techniques we used, KJ and RCA together with detailed minute writing and transcription worked well. The results were easy to understand for people reading the report afterwards, and we got a lot of information in a short time.

Conclusions

We started out with two questions – did our approaches work and did they differ in what we got out of them?

First of all – both the approaches worked quite well. The approaches were conceptually simple and had a low cost (the participants did for instance not need to prepare them selves) but even so; quite a lot of useful information came out of them. In Spacetec the results from the sessions led to a Contract Change Note to the customer regarding the amount of hours to be used in the project and useful input to the experience database. In InfoStream the results gave important input to a set of checklists that were under development. Thus, at least for SMEs, we do not need the extensive approach used by for instance Microsoft or Apple Computers. The companies involved found it easy to identify improvement opportunities and the participants found the approaches easy to use.

The two approaches differed mainly with respect to the types of information and improvement opportunities identified. We found that semi structured interviews worked well for a focused PMA while the other approach quickly lost focus. The other, open approach – KJ plus RCA – worked well in a catch-all situation and gave more surprises. An optimum solution would be to start out with an open PMA to identify all the issues and then follow up with focused PMAs on the most important ones. This will bring us close to a simplified version of the approach used by Apple Computers.

Acknowledgement

The authors want to thank the Norwegian research Council that has supported this

work through the PROFIT (PROcess Improvement For the IT industry) program and our colleagues at SINTEF and NTNU who, through discussions and comments, have provided many important inputs to this paper.

References

- [1] Collier, B. DeMarco, Tom and Fearey, P. A Defined Process For Project Postmortem Review, IEEE Software, July 1996.
- [2] Nolan, Andrew J, Learning form Success, IEEE Software, January / February 1999.
- [3] Cusumano, Michael A and Selby, Richard W. Microsoft Secrets, The Free Press, 1995, ISBN 0-02-874048-3.
- [4] Garvin, David. Building a Learning Organisation, Harvard Business Review, July / August, 1993
- [5] Kolb, David. Management and the learning process, in How Organizations learn (Ed. Starkey, Ken), Thomson Business Press, London, 1996
- [6] Straker, David. A Toolbook for Quality Improvement and Problem Solving, Prentice Hall International (UK) Limited, 1995
- [7] Scupin, R. (1997) The KJ Method: A Technique for Analyzing Data Derived from Japanese Ethnology, Human Organization, Vol. 56, No. 2, pp. 233-237
- [8] Jørgensen, Magne and Sjøberg, Dag. The Importance of NOT Learning from Experience. Proceedings of the EuroSPI 2000 Conference.
- [9] van Solingen, Rini, and Berghout, Egon. The Goal/Question/Metric Method, McGraw Hill, 1999. Japanese Ethnology, Human Organization, Vol. 56, No. 2, pp. 233-237
- [10] Mendoca, Manoel G. and Basili, Victor, Validation of an Approach for Improving Existing Measurement Frameworks, IEEE Transactions on Software Engineering, vol. 26,no. 6, June 2000.
- [11] Oivo, Marrku, Andreas Birk, Seija Komi-Sirviø, Pasi Kuvaja and Rini van Solingen: Establishing Product Process Dependencies in SPI, Proceedings of the fourth annual SEPG conference, June 7 - 10, 1999, Amsterdam, The Netherlands
- [12] Baird, L. Holland, P. and Deacon, S. Learning from Action: Imbedding More Learning into the Performance Fast Enough to Make a Difference. Organizational Dynamics, vol. 27, no. 4, 1999.

The following information is not part of the paper:

Appendices

CV for Tor Stålhane

Tor Stålhane was born in 1944 and became a M.Sc. at the Norwegian Institute of Technology, University of Trondheim (NTNU) in 1969. During 1969 to 1985 he worked at SINTEF - RUNIT, department for languages and compilers. From 1985 he worked on his Ph.D. studies and finished his thesis on software reliability in 1988. From 1988 he was back at SINTEF where he mainly worked with quality assurance and software safety and reliability. In 1997 he became professor in Computer Science at the Stavanger Polytechnic. In 2000 he became professor at the Norwegian University of Technology and Science in Trondheim where he today works full-time. During the latest decade he has been mainly working with safety analyses of software intensive systems and measurement based process improvement.

CV for Torgeir Dingsøy

Torgeir Dingsøy was born in 1973 and is working with knowledge management as a way to improve software development as a doctoral candidate at the Department of Computer and Information Science at the Norwegian University of Science and Technology. He is analysing the usage of knowledge management systems in two medium-sized software-developing companies in Norway, using a combination of qualitative and quantitative research methods.

He received his Master degree in computer science at the Norwegian University of Science and Technology in 1998, with a stay at Université Dauphine in Paris. He has published papers on knowledge management in software engineering, software engineering education, and on combining the artificial intelligence techniques case-based reasoning and data mining.

He is currently staying at the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern, Germany.

CV for Geir Kjetil Hanssen

Geir Kjetil Hanssen (born 1969) is a Research Scientist in SINTEF Telecom and Informatics. He is a M.Sc. (Cand.Scient) from the Norwegian University of Science and Technology (1996). He works within the computer science area covering various aspects of information technology. System architecture is one of his focus areas, and he has experience in systems design and systems engineering methodology and processes. Other focus areas are transport telematics and process improvement where he assists several Norwegian software companies in improving their internal software development processes. He is working with

messaging systems using XML based technology and a system for generating XML messages based on graphical message models (UML). He has experience from the IT industry, both as a systems engineer and developer and as a project leader.

CV for Nils Brede Moe

Nils Brede Moe was born in 1972 and became a M.Sc. at the Norwegian University of Science and Technology (NTNU) in 1998. His main research areas in the field of Software Processes Improvement include: Measurement based improvement, assessments and improvement on an organisational level. Other research areas are Human-Computer Interaction (HCI) and home care (health informatics).

He has been working on the research projects SPIQ and PROFIT, supported by the Norwegian Research Council.

Company description for SINTEF

SINTEF is an independent, non-profit research foundation based in Trondheim and Oslo, Norway. Our role is to encourage innovation and improve competitiveness in Norwegian industry and public administration. In doing so, we maintain close links with the technical Universities in Trondheim and Oslo, collaborating on projects, and sharing equipment and other resources.

With over 1800 employees and a turnover of NOK 1.4 billion, SINTEF is Scandinavia's largest independent research organization. It is organized into eight separate research institutes, covering all major scientific areas and industrial sectors. Refer to our web site www.sintef.no for further information.

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

- The SPIQ program – Software Process Improvement for better Quality. This is a national Norwegian program that is partly a national ESSI type project and partly a co-operation with Norwegian industry to increase the use of process improvement methods in Norwegian software industry.
- The QIS project – Quality Improvement in Scandinavia. This is an EU sponsored project that shall give help and assistance to PIEs in Norway and Sweden and market the concept of SPI to software industry in these two countries.
- The PROFIT program – PROcess improvement For the IT industry, which is a follow-up to the SPIQ program.

Company description for NTNU

As the name states the Norwegian University of Science and Technology, NTNU,

is a centre for technological education and research in Norway, with a solid foundation in the natural sciences. This tradition is interwoven with broadly based expertise in the classical university disciplines of the humanities, medicine and the social sciences.

The Department of Computer and Information Science is one of three departments at the Faculty of Physics, Informatics and Mathematics, NTNU. The department is currently located partly at Gløshaugen in Trondheim, Norway. Although modest in size (approximately 150,000 inhabitants), Trondheim is nevertheless the only Norwegian city found in Wired Magazine's recent overview of the world's 46 hottest spots of the global high tech network.

The department offers a broad selection of courses, covering most areas of computer and information science. Currently, the department employs approximately 90 faculty, staff, scientists and doctoral fellows. Approximately 30 of these are assistant, associate or full professors in permanent positions.

Integrating SPI and Assessment Models to Facilitate Measurement and Understanding

John Elliott
QinetiQ, UK

Introduction

Software process improvement (SPI) is part of the modern process paradigm for any software business. SPI is a business investment and its effectiveness needs to be monitored to ensure its contribution to business success may be understood and realised. SPI is fundamentally part of a systems problem – what does SPI interact with, what is its purpose, and what is an optimum architecture of a SPI system? Whilst SPI systems are an important part of the software industry, current SPI approaches do not adequately address the holistic issues that affect the effectiveness of SPI on business operations.

This paper analyses SPI effectiveness issues by taking a holistic view of SPI systems. An integrated SPI systems reference framework is postulated to capture the internal (supplier led) and external (customer led) business and technical relationships so that SPI effectiveness measurement and prediction can be based on firmer understanding. A SPI framework will enable criteria for ‘sub-models’ (e.g. process assessment, people competency) and their interfaces to be generated. Such criteria may be used to evaluate existing and emerging models, e.g. process assessment, to help to identify areas of improvement and to ascertain their contribution to SPI systems management. Furthermore, the framework can promote a high level strategy to drive model development to improve SPI effectiveness prediction and measurement.

The paper starts by clarifying the SPI problem domain by providing a systems perspective about SPI. This is followed by an outline of a possible holistic SPI reference framework model based on differing viewpoints. The paper concludes with a proposed way forward to advance a ‘holistic’ view about SPI to benefit the software industry and its customers.

Overview of the SPI Systems Problem Domain

SPI Context

All process improvement systems exist to contribute to the success of a business. They aim to be a vehicle for changing processes to solve reported 'production/service' problems and to seek gains in efficiency or effectiveness when providing products or services. SPI is no different apart from the focus is on the software production process. SPI is a part of a technical function that is within an overall scheme of business functions. The technical function is highlighted in the business context diagrams in Figure 1. This shows the multi-purpose of a technical function to support the internal business support functions as well as to provide the infrastructure for the customer related operations. Of particular note is the strong link between human resources and processes in delivering project operations.

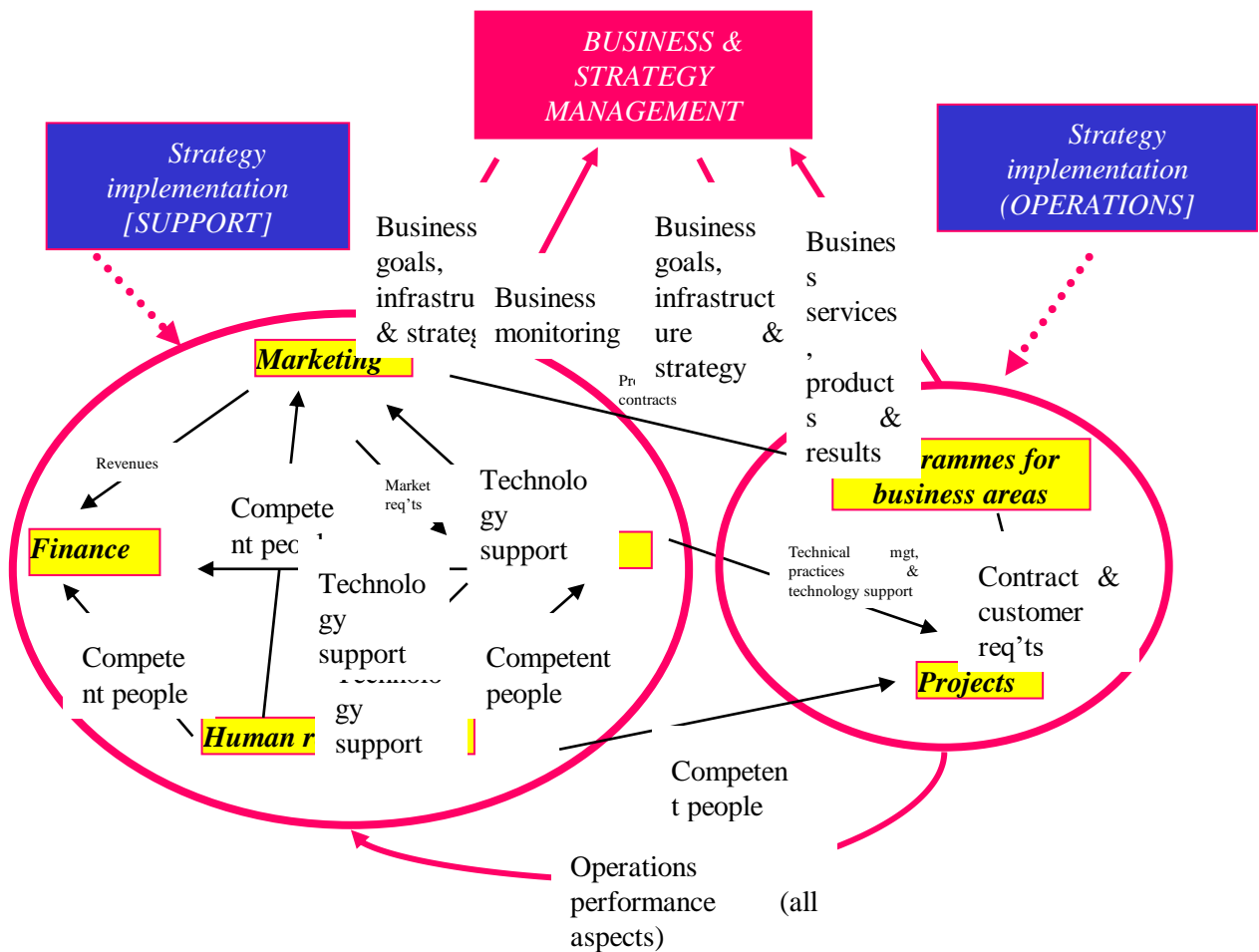


Figure JJE.1 Business Context for Technical Function

The technical function itself comprises of process related activities; process is intended here to represent management and technical processes and supporting methods and tools. Figure 2 elaborates the technical function by highlighting the definition, assessment and

improvement of processes in addition to components that capture ‘learning from experience’ and a ‘laboratory’ to investigate innovative opportunities. The ‘process defining’ amounts to standardising and guiding the business activities, often to be tailored, and applied in delivering products and services. Process assessment includes checking the process definitions and their implementation in projects using sets of industry recognised process criteria or benchmarking and performance measures or targets. The assessment results can influence where process needs improving. This is where CMM and SPICE methods may assist in relation to maturity or rating scales.

‘Learning from experience’ enables an essential source of evidence about the variety of internal application of processes within many types of projects. Such experience is a part of an overall repository for supporting process improvement and change decisions. The ‘laboratory’ offers innovation by initiating trials, evaluations and research issues as well as watching for external developments on experience reports and on technology or research results. The laboratory idea also provides the external evidence about existing or new processes to be used in making SPI decisions; for example, what do we know about the likely effectiveness of method X in our business environment.

The process improvement system can thus be seen to be influenced from many sources (assessments, experience, innovation) but the most common source will be the direct identification of process problem that need solutions. These problems may also be a result of an unsuccessful project and the causes are believed to be partly due to unclear or inappropriate processes, in addition to implementation issues, e.g. poor training, new tools or inconsistent u

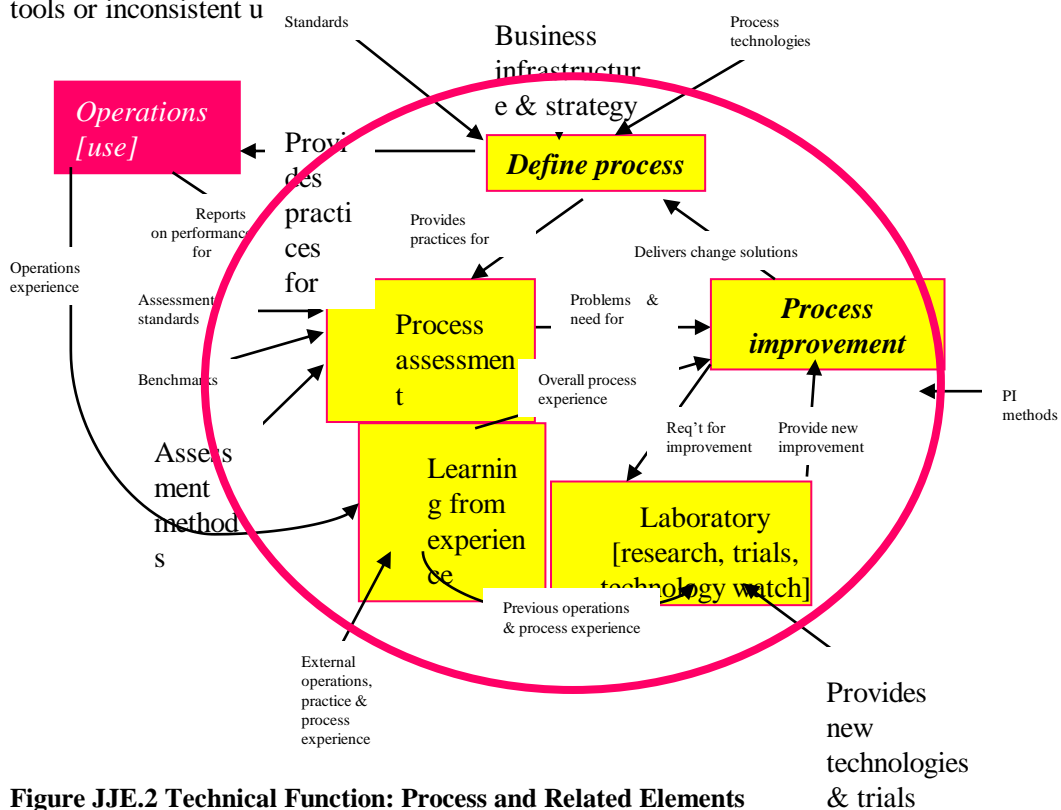


Figure JJE.2 Technical Function: Process and Related Elements

Finally, if we consider what SPI systems need to do and what kind of decisions are involved, the context for an analysis of how to approach SPI effectiveness will be properly set. As indicated, SPI functions receive demands for change and each demand

needs to be studied and actions taken. The SPI function reviews the needs and opportunity-costs-benefit of potential process changes, and this may involve initiating trials, evaluations and even research studies. The SPI processes involve:

- ? Understanding why a change is needed, e.g. problems/symptoms, context, business changes.
- ? Defining the objective of any proposed change.
- ? Identifying the different change options to meet the objective.
- ? Evaluating the different options to identify costs and benefits to the business.
- ? Selected the option to be tested further to assess its impact in real projects, e.g. trials or experience reports.
- ? Conduct trials and evaluations providing evidence about the process change.
- ? Review the evidence and either reject the change, or request more evidence from trials etc before 'roll-out', or accept the change and incorporate into standards and training programmes.

The crux of a successful SPI function is to be able to predict the impact of a process change on the business operations. In other words, how will any change affect the success of the whole business through enhancing success in the software projects? The whole business success means looking at the overall high level business performance, maybe based on a balanced set of indicators addressing financial, customer, process and people aspects. The issue is how can these indicators be derived from an analysis of process change. The software project success means exploring many 'balanced' indicators such as the actual duration, time, cost, human resource factors, process quality compliance and product quality attributes.

SPI Problems

Now the context is set the generic SPI problem is elaborated. Currently, the SPI systems have limited ability to predict their value for money. Many references have reported great benefits of SPI but these experiences tend to be from large corporations without providing any information or evidence on how the process change decisions were made, what process evidence was used and how accurate were any process change effectiveness predictions. In general, many SPI systems operate within limited resources and access to internal and external process experience and knowledge. Hence, there is a need to provide a holistic framework within which there are different 'integrated' method components; some key methods (e.g. process assessment) already exist with different degrees of maturity but there has been limited advancement on their integration.

In effect, SPI systems help manage and control software process change but usually in terms understandable to a software practitioner, not in terms relevant to a business or customer manager. Future SPI systems should relate more strongly to business issues whilst satisfying the needs of the SPI decision makers (leading to an investment in SPI or an implementation of changed software processes).

A key difficulty in the software industry is a lack of data and analysis, and a commercial reluctance to share data or results. However, much data may be easily available for assessing SPI effectiveness without being realised. Despite the data difficulty, there is much information available about software engineering in many publications, even available across the Internet. However, without extensive 'data mining', it is likely that much hidden understanding about software process effectiveness and experience will not

be realised. Ideally, such understanding needs to provide key arguments about the effectiveness of different processes or methods within a context setting, even if the argument is based in subject as well as objective information. The existence of a SPI framework will be a first step to harness such arguments in an evolutionary manner. Without an agreed SPI framework, there does not exist any standard, even 'de facto', to guide any shared SPI analysis. This is an important rationale for such a framework – it will encourage an external 'learning from experience - body of knowledge' to be captured and suitably shared among the software industry. SPI cannot be performed effectively without an internal or external repository of evidence from which SPI arguments, to justify a proposed process change, may be developed.

Most of the discussion on SPI context and problems to date has inferred the emphasis is on SPI effectiveness for a software supplier business. However, any holistic view of SPI also needs to account for a need to evaluate the benefits of a supplier's process change on customer businesses. This may include improved evolutionary or adaptive ways (maybe new supplier processes) of helping the software customer define and use the new software to lead to a successful operation.

If the market were safety critical systems, and a software organisation wanted to grow a capability and reputation in advanced software engineering, e.g. mathematical proof techniques, then any new processes will need to impact on major product quality activities and specialised assurance techniques to evaluate safety and risk. Any SPI actions will need significant evidence and innovative studies before confidence, including a customer's viewpoint, will be attained in the organisation's ability to create safety related software.

A holistic approach to solving the SPI effectiveness problems is expected to provide any understanding about the relationships involved, to determine any 'cause-effect' links and also to predict any emergent (undesirable) issues that may arise when applying and integrating different methods. The different method components and their interfaces will lead to measurable attributes that will form a basis for future SPI effectiveness and process change predictability. The result will be a basis to achieve traceability from SPI to business indicators and a means of evaluating which SPI decisions will lead to acceptable business benefits.

Thinking strategies to date

Some authors have suggested that an almost fanatical focus on process in the 80's and 90' is the root cause of a second potential 'software crisis'. The rationale is that software process thinking and SPI, whilst good analysis tools, do not wholly address the wider influences on software business and project success. Process thinking provides a valid baseline for assessment and improvement. So assumptions about the benefits of different processes in a world where there is rapid software technology changes means that the relatively static world of process management needs to be examined in a wider context to seek more dynamic and flexible solutions. Also, the need for process variability is heightened, as increasingly distributed and complex software systems are required in shorter timescales, yet still with challenging quality requirements.

The process thinking approach also implies assumptions about 'statistic process control' that tend to be rather invalid when it comes to innovative software design and production. To strengthen such assumptions, ideally a theory about software is needed to strengthen the foundations of current or evolving process based methods; no theory is evident at

present. Much of the process revolution has been based on logical and empirical studies and beliefs to control and limit product quality variations. Despite these potential needs to build on current process thinking, there still needs to be an established SPI and software process effectiveness framework with a business decision-making context. Apart from process thinking in the software industry, there are other 'thinking' paradigms that have influenced the evolution of software concepts, all designed to prevent any notions of a continued software crisis:

- ? Business thinking – in general, this has emphasised the structures of business [functions, process and organisations] to deliver services and products. The general business community has adopted process re-engineering, business excellence (Business Excellence Model) and measurement regimes (Balanced Scorecard) to rationalise and improve an organisation's focus. These ideas have been adopted in the software industry to enhance the management approach.
- ? Quality thinking - the focus on quality dates back many years and is well documented. The impact was to define what a business does, or should do, and to focus on how it achieved satisfying customer requirements. This means that businesses have adopted 'quality' practice to follow good generic principles to systematically define requirements, plan for its achievement through design and implementation and incorporating quality control practices.
- ? Product thinking - one of the problems with process thinking is the difficulty in explaining and predicting product quality due to the complexities involved. Product thinking is a special focus on the attributes of software to determine 'fit for purpose' quality, often related to 'levels of control or integrity'. Each level relating to different degrees of perceived risk in terms of the likelihood and impact of inadequate product quality. Different product assessment methods have been developed to examine different product attributes. Extended product thinking has been developed to address reuse, COTS and product certification.
- ? People thinking – there has been a number of initiatives to support the individual processes (Personal Software Processes (PSP), PISPI). In addition, there has been some various attempts to capture and assess skills and competence levels for software project personnel [British Computer Society- Industrial Structure Model, EU CREDIT project]. Apart from that within PSP and PISPI, there have been limited attempts at integrating skills and processes for individuals. Current team-working studies are exploring the communication issues when applying the SPICE process reference models (EU TEAMwork project).
- ? Measurement thinking – in the 80's and 90's, there were many developments in the theory and practice of software metrics to address process, product and project measurements for assessment and for development control. Within other EU studies, complexity analysis for products and process has been an ongoing development to indicate difficulty levels for costing and quality assessment purposes. Also, there have been a number of methods for addressing the measurement issues about business (Balanced Scorecard, SPI (EU 'ami' project) and process maturity (CMM)).

The key issue is that these different 'thinking' paradigms may have generated parts of a

potential holistic SPI framework that does not yet exist; any future SPI framework will integrate the different thinking approaches. The next section explores how such a framework might be progressed. It would be not be right to omit reference to valiant attempts at addressing the holistic issues. One to mention is the EU PICO project where a spiral based leverage model was developed to show how a relative small changes to a process provide leverage effects might generate bigger impacts at a business level. Other attempts by the EU Bestregit project that provides a goal oriented approach to SPI so that process changes are linked to business goals. Also, the continued balanced scorecard development has been extended to the IT sector, referred to as BITS. There have been some studies to integrate selected methods, such as BITS, Business Excellence Model and SPICE. Related studies in setting frameworks for 'customer satisfaction' have emerged [EU REJOICE project] and these have generalisations that may help a structure a SPI framework development. These attempts at integrating existing methods have all developed understanding to be built upon but they need to address the wider issues.

SPI Reference Framework

General

The previous sections have discussed the SPI systems context, highlighted problems areas and described current compatible 'thinking' paradigms.

This section describes, in outline, different views of a 'holistic' SPI reference framework that will be a 'meta-model' of SPI systems to strengthen the scope of integration and evolution of different methods and models about business, software development, process and product assessment, and improvement activities. Furthermore, any SPI framework will highlight different system components that impact on process and business effectiveness in some manner; for example, a people component can be included to help capture process–people interactions.

The benefit of a holistic framework will be to encourage a wider focus allowing the business impact of SPI to be assessed systematically and systemically; this impact will address supplier as well as customer businesses. After all, if the customer cannot justify the cost of new software then the software industry is in real trouble!

The SPI framework should accommodate the inclusion of increasing knowledge, understanding and related measurements as they evolve. The test for such a model will be its ability to be used to tailor and adapt to new and dynamic developments, initially designed to enhance the software industry.

Account will be taken of previous models and attempts to consider integration issues. Many ongoing developments of assessment and improvement models are founded on enlargements (i.e. adding systems concepts into software engineering) to capture any evolving, and often limited, understanding (even beliefs) about those factors affecting the production of quality software products. This framework will facilitate such enlargements in understanding.

Overview of a SPI Framework and its Component

The purpose of a SPI reference framework is:

- ? To capture the essential elements of a SPI system to enable the business benefits to be ascertained.
- ? To provide an understanding of the influences and impacts associated with SPI decision-making.
- ? To identify and describe the major components of a SPI system.
- ? To facilitate in the development of culture to share SPI results.

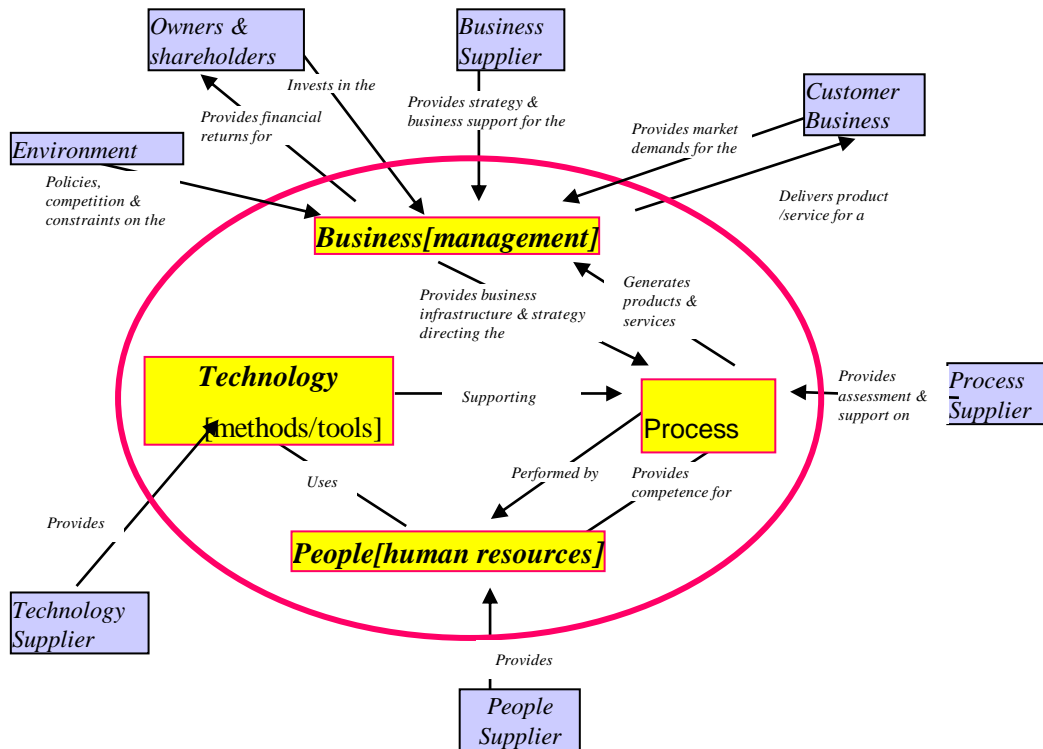


Figure JJE.3 SPI Reference Framework: Component View

Based on the understanding presented in the SPI context description, the main components of a SPI reference framework, as in Figure 3, for a generic supplier business are described. This represents all the areas that SPI needs to address through different focussed methods but the framework will explore the interfaces to pursue their interactions. The framework will also capture the overall SPI and any emergence properties arising from the combination of these parts.

The framework components to be addressed are:

- ? **Business [management]** – this captures the business context and its functional structure emphasising the management of the business and all its projects, of its supplier-customers interface [customer management] is sufficient understanding to enable some mapping with the use of processes and associated technologies by people to be established. This distinguishes between ‘enabling’ activities and ‘delivering’ business results. Different internal (processes, people and learning) and external perspectives (financial and customer led) need to be addressed.
- ? **Process** – this is the whole life and support management and technical activities as necessary to generate and deliver the services or products to customers.

- ? Technology [methods, tools and environments] – this is the supporting development or project environments and techniques to perform the processes (modeling techniques for requirements engineering)
- ? People - this is the characterisation of a software workforce (roles, skills and competence), and their role and social interactions within teams.

Layered View of a SPI Framework

These components are the main areas that must be studied to ascertain SPI effectiveness. Another presentation is to separate the influence of these components as different layers. Of course, each layer represents a system boundary, and collectively all layers will have interacting ‘system of system’ layers; the result is that the impact of a process change within the infrastructure layer will be traceable to the business layer to see impact of such change on the different business elements or attributes. Also this framework will allow the impact of a process change to be predicted and argued based on available soft or hard (subjective or objective) evidence. It is the ability to argue about such process or technology changes that is critical. The framework is designed to achieve this impact. The layered approach is applicable in SPI analysis to decide what process change to explore or implement, as well as deciding the overall impact of the SPI function, part of a technical function, as in Figure 2. The elements of each layer are now outlined, also shown in Figure 4:

- ? Infrastructure layer: Processes (whole life cycle and support processes) and supporting technology (methods and tools) and human resource practices (skills development, competence levels).
- ? Project layer: People (allocated human resources, roles and skills and competence matching), process selection and tailoring, technology selection, product development project and technical assurance (acceptable use of processes, technology, demonstrable ‘fit for purpose’ product quality).
- ? Business service layer: customer-supplier business interface, customer service attributes (adaptive, evolutionary, team-working) and business assurance (meet needs and defined requirements).

This layered approach uses some of the structured developed by a reference framework designed for designing customer oriented evolutionary processes [EU REJOICE project]. This has been generalised to add more structure into the layered model designed to show traceability.

competence for

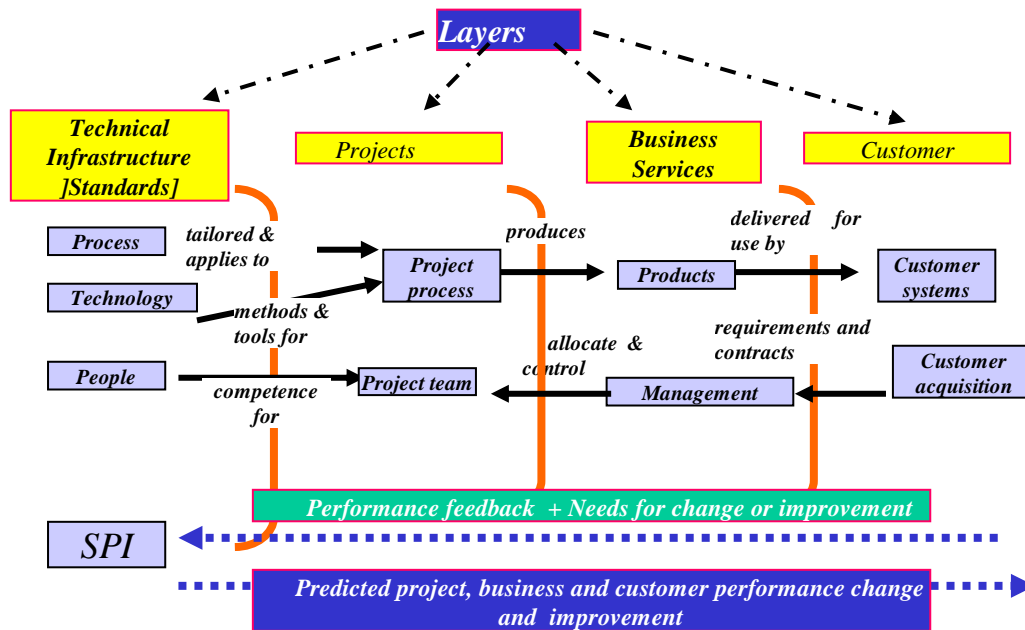


Figure JJE.4 SPI Reference Framework: Layered View

Dimension View of a SPI Framework

An alternative ‘dimension’ view of the SPI framework has been created based on generalisations of the customer reference framework [EU REJOICE project]. SPI effectiveness will have many different business variables about financial, customer, process and people perspectives. This framework will be interpreted for variables, such as ‘customer satisfaction, in terms of four dimensions [main concept areas], attribute areas [measurement areas] and qualifier [influence areas]. The value of this additional view provides a structured means of developing the SPI layers view; for example, the financial impact of a process infrastructure change will need to be analysed in terms of certain dimensions, attributes and qualifiers. This generic dimension, attribute and qualifier views of SPI reference framework is captured in Figure 5, are now outlined.

The dimensions and attribute areas include:

WHY/WHAT [BUSINESS] DIMENSIONS

- ? Goals - the business needs or aims [attribute areas: business context and change strategy, market conditions, user requirements]
- ? Requirements - a solution to meet the needs [attribute areas: system definition, quality and risk levels, project or people competence criteria, technology reuse, external interfaces]

IMPLEMENTATION [TECHNICAL] DIMENSIONS

- ? Approach - how the solutions developed [process and people emphasis] [attribute areas: process definition and criteria, people competence, quality techniques]
- ? Result – the demonstrable ‘certified’ and fit of purpose quality’ solution [attribute areas: fitness case and argument, product quality demonstration and evidence with supporting process and people evidence.]

This set of dimensions [and attribute area] view can be interpreted to represent SPI

system development or software product developments. For SPI systems, the business dimensions sets out the SPI system needs. The technical dimensions sets out the approach and results of developing the SPI system. Each dimension has many attributes to the basis of future measurement schemes. In addition, the attributes of each relationship between dimensions for each variable will capture the degree of ‘satisfaction’ of ‘influence’ between the dimensions.

The qualifiers for each of the SPI effectiveness view (e.g. financial impact) will require additional perceptions that influence the degree of impact on that view. These qualifying perceptions include:

- ? Assurance that business goals met and the technical ‘fitness’ of solution
- ? Team working arrangements will affect how the business and technical dimensions are managed.
- ? Evolutionary approaches must allow for continuous improvement and to respond to needs for change
- ? Adaptability is an important need to respond to changing business needs to ensure the results are relevant and provide the expected business gains.

The assurance elements will qualify the dimensions of ‘approach’ and ‘result’ to argue fitness and need satisfaction. The remaining team-working, evolution and adaptability qualifiers all support the dynamic satisfaction of business need through constructing relevant solutions.

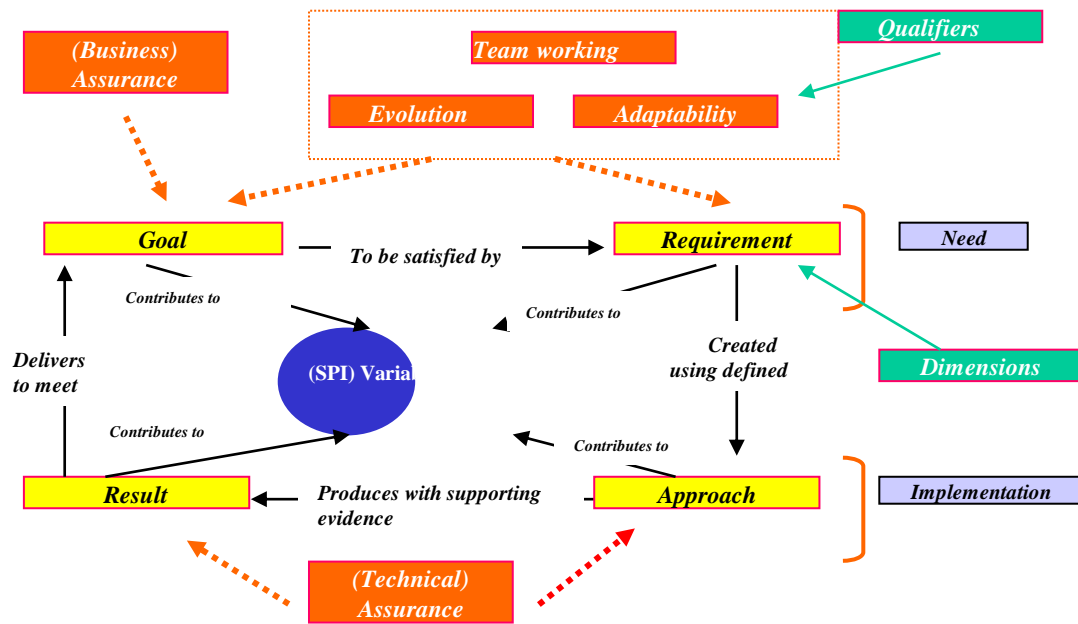


Figure JJE.5 SPI Reference Framework: Dimension View

Summary of SPI Framework

The SPI framework idea has been described in terms of its purpose, components together with a layered and dimension views. All these ideas have been presented to stimulate further discussion and development. The main purpose of the framework is to provide an evolutionary means of directing the establishment of an integrated framework to support SPI systems management and effectiveness assessments.

Way Forward

This paper has set out a systems view about SPI to accommodate more extensive analysis to assess different project and business effectiveness. To address the SPI effectiveness problem, a SPI reference framework has been proposed. This framework needs to be further developed. However, a number of different and related views about an evolving and dynamic SPI system have been presented to stimulate discussion on the SI effectiveness issue. .

In support of a holistic SPI framework, the following steps are needed:

- ? Elaboration of the SPI framework components and their links using a layered and dimension analysis.
- ? Using the SPI framework, develop criteria to be satisfied for constructing an useful framework.
- ? Using the criteria, evaluate the ability of existing method/framework components.
- ? Using the criteria, determine future research needs to create more integrated and improved method/framework components.

The major step will be to elaborate the SPI framework through further research and analysis. The benefit of a framework will be an enhanced understanding of SPI systems and also provide a firm basis for evaluation and measurement of software processes.

Acknowledgements and Disclaimers

The author would like to acknowledge the internal QinetiQ (previously Defence Evaluation and Research Agency) funding on the general concept studies and the external European Commission funding of many projects, one of which particularly stimulated the ideas in the ideas in this paper, REJOICE, Project 23893.

The views expressed in this paper are entirely those of the author and do not represent the views, policy or understanding of any other person or official body. Any feedback, expressions of interest, or just further details can be requested from the author at the QinetiQ, Systems and Software Engineering Centre, Malvern, UK; Tel: +44 1684-895161, E-Mail: jjelliott@qinetiq.com.

Session 6 - SPI and Processes

**Session Chair:
Tim Hind, Axa, UK**

ISO 9001 as a process improvement tool; Is ISO 9000: 2000 the last chance for the standard?	6-2
Comparison of Software Process across Small Companies	6-18

ISO 9001 as a process improvement tool; Is ISO 9000: 2000 the last chance for the standard?

Philip Wain
QinetiQ, UK

Introduction

ISO 9000 has been with us for almost 14 years now, with the TickIT scheme following closely after, and industry's experience with it has been a profound disappointment. It has not delivered the promised process improvement benefits and attacks on its integrity have been growing more damaging. QinetiQ, then operating as DERA, was approved to ISO 9001 and TickIT in 1994 [5] and our experience has been typical of many. This paper attempts to explore some of the reasons for the failure of the ISO 9000 family to deliver process improvements in software despite its continuing high popularity particularly in the UK. The paper makes particular reference to actual DERA experiences in implementing the standard and our attempts to use it as a vehicle for process improvement in the software arena. We take a critical look at the changes in the 2000 edition and the accompanying guidelines for software in the draft ISO 9000-3 and the TickIT Guide version 5.0, [7] to examine whether it has the capability of delivering process improvement.

ISO 9000, a brief history

ISO 9000 was born out of a desire to have a civilian standard that described a system for controlling those elements of design and manufacturing processes that

led to mistakes, rework and failures in product quality. The military industrial complex of the USA, UK and other NATO countries already had such standards in the decades following the Second World War. Those standards, typified by NATO's AQAP 1, and 13 for software, had been evolved and developed because of a twofold need; for frontline forces to ensure the efficacy and performance of munitions in the field, and for manufacturers to prevent the explosions of munitions during production. Those standards had their roots in control regimes that had grown up with mass production between the World Wars, and had developed, in the main, independently of the Quality ideas of post-war American and Japanese gurus such as Deming, Juran, Crosby and others.

The British Standards Institution published civilian guidelines on Quality Management in the mid-seventies, and the first assessable standard, BS 5750 in 1979. BS 5750 became the "straw man" for ISO 9000 which incorporated a lot of the ideas into its publication in 1987.

The military standards were essentially 2nd party standards; they were applied in contractual relationships between defence contractors and their customers, Government Defence departments, with the customer having the right to audit the supplier's arrangements for fulfilling the requirements of the contract. Auditors were agents of the customer directly, and were therefore there to police the implementation of the standards with some rigour. ISO 9000 was designed from the outset to be a true 3rd Party standard, in which the auditors would have no direct contractual involvement with the company being assessed. One of the standard's stated aims was to reduce the auditing burden on companies with the standard, so instead of a number of different customers auditing a firm, one certification body would audit and award a certificate, giving potential customers confidence in that firm's ability to deliver.

Before its adoption by ISO, the auditors of BS 5750 were mostly from the BSI, and they behaved similarly to the Government auditors, being quasi-Government themselves. The eighties saw a move away from Government-run services and the certification regime that was set up to support ISO 9000 was entirely profit oriented, including that run by the BSI.

The TickIT Scheme

Experiences in assessing software companies to ISO 9000 showed that domain knowledge of quality auditors was vital. Few software professionals (who were paid significantly more than the certification bodies could afford at the time) were attracted to this line of work. The disappointment of the software industry in the UK with the quality of auditing in their domain led to the TickIT scheme, the first industry sector scheme for assessing ISO 9000. It later spawned many others. Its main drive was to improve the selection of auditors with software expertise and to improve their levels of remuneration, to encourage software professionals to do that kind of work.

The TickIT scheme authors also felt that the ISO 9000 standard was too biased

towards heavy manufacturing/engineering, and so resolved to publish interpretive guidance in applying the standard for software development companies. The guidance was taken from the UK's National Computing Centre's STARTS Guide containing software engineering good practices and the BSI's Quality Auditor's Guide for software given to their own assessors. At about the same time, TC 176 was already working on such guidance at the ISO level. A draft of ISO 9000-3 was circulating when the first version of the TickIT guide was being published, and so for the first few editions of the TickIT guide, the ISO 9000-3 guidance was included verbatim.

The rise of ISO 9000

Adoption of the standard in the UK was rapid thanks, in part, to a common misunderstanding of the purchasing requirement in clause 4.6. The standard required the adopting organization to assess the capability of its suppliers to deliver on quality. Many took this to mean that they should require their suppliers to have ISO 9000 too. Many of the big procuring companies adopted this policy and were soon forcing ISO 9000 certification inappropriately on all kinds of firms, who had to comply if they wanted to retain the business of these important customers. When the UK MoD switched their contract quality requirements to the civilian ISO 9000 in 1993, the process intensified. Other parts of the UK government followed suit. Also, because of the adoption of ISO 9000 as a Euronorm, many firms outside the European Union began to believe that failure to adopt ISO 9000 might mean they would be excluded from the European market so began to adopt it too. A small but significant number of US and Japanese firms began to appear to be led on Quality by the UK!

Today, the ISO 9000 standard remains the single most popular selling British Standard ever published.

ISO 9001: 1994, TickIT and Process Improvement

Potential for Process Improvement from ISO 9001: 1987 and 1994

Those of us who were working in the Quality field in the late eighties when ISO 9000 was first published, embraced the new standard as a vehicle for delivering the benefits of total quality espoused by Deming and the rest which we had read about and been persuaded by. Much of the rhetoric involved in selling the standard to us used the concepts of total quality as its rubric. For us, it had the main attraction that because senior management needed external approval, finally, we were being listened to in a way that had not happened previously. Many of us used the excuse of the need to comply with the standard to encourage our employers to adopt quality practices that they otherwise would not have entertained without clearer quantified justification. We could not believe our luck, and relatively junior managers, myself among them, found themselves with unprecedented influence over the management of our organizations, yet many of our convictions were based on little but faith. The fact that the earlier versions of the

standard were written in terse almost legalistic statements of general requirements meant that the standard could be our *tabula rasa* for whatever quality guru we, as individuals, supported and believed.

The fundamental first step in process improvement, we believed, was that to improve a process you first had to understand and define it and assess its current performance. Processes which depend on people's behaviour for their correct execution must be described in such a way as that everyone executes the process in exactly the same way; a means of controlling at least one important source of variation. No improvement was possible until there was a baseline from which to improve.

Unquestionably, we believed, the ISO 9000 standard required that processes were defined, because documented procedures were required for all the elements of the Quality Management System. We believed that the interaction of these processes would also be covered because the ISO 9000 standard described a model of design and manufacture including all elements that affected quality. There was an inherent end-to-end process implied by the twenty clauses in the standard, from Contract Review at one end, through design planning, design and design verification, production planning and process execution, testing and inspection, bonded stores and despatch, even servicing of equipment in the field. Inputs of people and raw materials were covered by supporting clauses, as was the use of consistent measurement equipment and tools.

Once the system was fully defined, monitoring processes such as internal audit and analysis of customer feedback would ensure that the system would become more efficient and effective over time.

Those of us who were committed to and understood process improvement could persuade our companies to adopt enlightened structures involving process definitions, systems for controlling variation within those processes and measurement schemes to establish process performance and to improve upon it (although the measurement schemes proved to be the hardest to implement). At the time, I believed, naively, that this was the only way that the standard could be implemented. I was astonished to discover, having become a registered TickIT Auditor in 1993 and starting work for a leading certification body, how little a firm could do and still be compliant with the letter of the standard.

Experiences in DERA

As we can see from elements of ISO 9000's history, in very few cases was ISO 9000 adopted by firms because of its potential as a process improvement tool, but because it was being imposed upon them by an important customer. DERA was in this position in 1993, when the UK MoD published its NACNOC policy, standing for No Appropriate Certification, no offer of Contract and under DERA's terms of Business arrangements with MoD, the policy applied.

Once the decision inside the organization was taken to adopt the standard and go for external approval, management imposed very tight timescales by when this was to be achieved. Funding for the ISO 9000 implementation was forthcoming and the staff set to work.

DERA began to implement ISO 9001 and TickIT across the organization, in a piecemeal fashion, with individual operating units responsible for creating their own quality management systems and seeking their own certificates of approval. The exception to this was the software practices which were developed for use for the whole of DERA using corporate level funding. The operating units mostly chose to adopt this set for their own use, although there were one or two who went their own way on software too. After the early experiences with certification there was co-ordination of the approvals from the Centre and the process was over by late 1995.

The speed by which the activity was carried out did not leave a lot of room for widespread consultation and selling of the benefits to the business regime of certification. Rather, heads were placed on the block to concentrate minds and procedures were taken off the shelf or from other organizations and re-badged in order to get a fully functioning quality system in place as quickly as possible. Practices for software development were adopted from the ESA Software standards published as PSS 05.

DERA was fundamentally a research organization, and from the beginning there were difficulties in interpreting a manufacturing engineering standard for use in that environment. In addition, the personality types found inside the organization were not well-disposed to having detailed controls imposed upon them. At the time there appeared not to be very good mechanisms of accountability in terms of whether research funds were being used efficiently and effectively, whether good value from those funds was being derived. All the staff was aware of the need to use allocated funds carefully, they were, after all, public servants, but the mechanisms for ensuring this were simply absent. I think it is also fair to say that there was not a good understanding of the research process, which appeared to be undefined, *ad hoc* and not monitored well. As for software, few of the scientists developing software inside the organization were in fact software professionals, but the use to which the software was being put was for demonstrations and simulations and therefore apparently low risk.

However, by a mixture of coercion, hiding of problems, looking good on paper, management of the external assessment process and luck, DERA steeled itself for external assessment. They were in many cases astonished at the ease with which they came through the assessment process and gained approval.

Once approval had been gained for all parts of the organization, the people within the quality infrastructure, that had been created to help achievement of the standard, began to try to use this hastily created system to deliver the quality and process improvement benefits that the rhetoric suggested should be forthcoming. However, now that certification had been achieved, management attention moved elsewhere, and in successive years, the quality efforts began to be pruned. Centralization was an obvious way in which savings could be made and the decision was taken to merge all the disparate quality systems into one Business Management System for the whole organization. Procedures were to be simplified and generalized so that they could apply everywhere. The BMS was completed in

early 1997.

Staff, who had been largely sceptical during the drive to certification, but who had largely bowed to the pressure to conform and played the game at least until after the auditors had been, now began to return to their old undisciplined unrecorded way. This behaviour intensified when it became clear that the surveillance visits by the external assessors did not seriously challenge the system to any great extent and the staff found it relatively easy to accommodate the external auditors for the extremely short time they were there. Internal audit arrangements were no stricter as the durations, coverage and depth for which had been set to match closely the times duration and depth for external assessments. Also the requirements in the standard on corrective action were not properly applied, in that it was often possible to close an audit non-compliance on the basis of having corrected the fault observed but not its cause.

The extensive autonomy of operational units made it difficult to persuade projects to participate in a corporate level process improvement activity. Although the BMS procedures required projects to complete metrics returns at strategic intervals and at closedown, there was no concerted way of knowing which projects were live, so no way of knowing if returns were to be expected, or missing when they did not show up. Figures showed that improvement suggestions from staff were taking an inordinate amount of time to be addressed. Although nominated staff had been found to own each BMS procedure, few of these staff had either the time or the funds to implement the improvements. Again, the exception to this was the centrally funded and strategic software procedures. However, due to a general disillusion with the process, involvement in the improvement proposal system meant that the numbers of improvement suggestions gradually dwindled even in the software domain, as staff learned that the improvement staff were generally unresponsive.

The above experience looks like the implementation of ISO 9000 in DERA was both shallow and cynical. But it must be stressed that, although DERA had some unique attributes that made implementation of the standard particularly difficult, the DERA experience is typical of many firms who, having had the requirement for approval to the standard imposed upon them, do the bare minimum to achieve approval and then ensure that maintaining approval troubles them as little as possible.

In fact QinetiQ's commitment to Process Improvement is very high, and in many parts of the organization, alternative infrastructures have been, and are being, created to deliver the benefits that ISO 9000 has failed to deliver. Although these are not in contravention of ISO 9000 they are not required by it either.

Why did the ISO 9000 standard fail to deliver Process Improvements?

The ISO 9000 infrastructure consists of three main elements: The Standard itself, The Certification system and the organizations applying the standard to their own companies and imposing it as a contract requirement on their suppliers. I believe that all three of these elements were flawed.

Flaws in the Standard

I believed at the time I was involved in implementing a software quality system to ISO 9001 that the documented procedures required by the standard were almost synonymous with processes - the one described the other. In fact, a critical look at the standards requirements reveals that merely being required to have a documented procedure for everything does not automatically lead to understanding of business processes and their interactions at all. There was little consistency across industry as to the extent of detail and coverage of a procedure. The 1987 version included the concept of a work instruction, but this spawned more confusion than clarification and the term disappeared in the 1994 version.

The Word “Process” appears in the context of the manufacturing process only, and conjured up images of factory process, heavy machinery and operators turning dials and pulling levers. Business processes were clearly not the intention of this clause.

Seddon [1] notes the tendency of ISO 9000 to engender internal focus when it refers to procedures instead of processes. Processes are focused on delivering on the process purpose, which is to achieve successful outcomes for the process stakeholders. Procedures are on the other hand focussed on locking down behaviour into a consistent operating environment. Seddon [1] uses the example of a printer’s quotations system, which, although fully documented, recorded and monitored, was causing the company to lose market share because customers were switching their business to a company who could respond with quotations faster. Whilst ISO 9000 1987/1994 did not prevent procedures from reflecting business processes, it did not encourage it either.

The word, “Improvement” does not appear anywhere in the requirements of ISO 9001 1987 and appear once in the 1994 version clause 4.1.2.3, where it says the management representative should report on the performance of the quality system “as a basis for improvement of the quality system”. In fact the objective of the corrective action and preventive action clauses were to change the system only to prevent mistakes and not to improve efficiency at all.

There was a heavy emphasis on documents and records in the standards. The oft repeated mantra used to describe the philosophy behind the standard, “say what you do, do what you say” (and make a record of what you did so that you can prove you did it) led to a tendency to over-document. Once the procedure stated something had to be done then a record had to be created to leave an audit trail, often to no other purpose. Seddon[1] speaks of doing the work, then writing about it. Whilst many records were valuable to those trying to assess process performance, many were there simply to show that a step in the procedure had been taken, when, had that step been missed, there would have been no serious consequence. Naturally, intelligent members of staff at first questioned this need, then grudgingly complied for a short time, then lapsed into cynicism and despair, then

simply ignored the requirements.

Some quality consultants have noticed that by reducing the detail in procedures, there is a consequent lower need for records and therefore less room for external assessors to take issue with implementation. Some have taken this to an extreme degree such that the lack of detail in the procedure makes the procedure completely useless in supporting the activity.

The terse nature of the first versions of the standard have already been referred to in allowing enlightened quality professionals to persuade managers to adopt sound quality practices which will lead to quality improvements. However, the downside to this flexibility is that less enlightened staff with quality responsibility can equally persuade managers to adopt inefficient practices, which will not lead on to improvement. Instead they reflect the Taylorism of the Fifties and Sixties, during which time many of the more elderly quality personnel both internal to the company and those working as consultants, cut their management teeth. Both strategies have proved to be equally successful at achieving approval to ISO 9000.

The Certification System

As has been said the intention of the ISO 9000 standard was that it would primarily be a third party assessment standard, with firms assessed by independent certification bodies who would sell their services to these firms for profit. The certification bodies, if they wished to offer accredited certifications, would have to be accredited by an overseeing authority; in the case of the UK this is UKAS, formerly NACCB. This body was initially attached to the UK DTI and so was quasi-Governmental, but it too now is a profit making organization.

Certification bodies do not operate to ISO 9000 themselves but to EN 45012, a European guideline that has similar provisions to those in ISO 9000 but not all.

There was an expectation in companies applying for certification about the costs of the process and early take-up was slow. The BSI, when auditing against BS 5750 in the early Eighties had charged administration costs, but had not built in huge profit margins. To operate a service to those scales of charges, external certification visits had to be brief and shallow to make it affordable. Furthermore, the nature of the certification process was such that problems could be weeded out at early assessment visits with subsequent assessment visits focussing on only those items that had failed last time. With the ability for management to then focus its attention on a smaller and smaller number of problems, the process was designed to ensure that firms achieved registration eventually. It was of course in the interests of the certification bodies to make sure their clients eventually passed, because subsequent surveillance business ensured a steady income stream for them.

Initially, certification bodies policed the assessments of their customers to ISO 9000 as if they were an independent authority, even though their services had been commissioned by paying customers. Market conditions allowed this, because demand for certification in the late eighties early nineties far outstripped supply and competition between the

existing accredited bodies was not an issue. I was working for a certification body in 1993 when I noticed a sea change in attitude, brought about, in part by some very big companies who began inviting certification bodies to tender for certification business. I watched light bulbs going on over management heads as they realised their place as a mere supplier in the marketplace. They hadn't thought of their customers this way before and began to consider what it was their customers actually wanted. They arrived at a decision that I think was a key seed of destruction in the whole edifice of independent certification.

They concluded that the service they were offering was a value added review to help their clients improve their management systems. Their whole focus was then turned on how to give better value during audits and offer help, in interpreting the standard, and suggesting implementation strategies. Assessor/Auditors were encouraged not to look for problems in implementation but to look for evidence of compliance, only the absence of which would warrant non-conformances being raised and a failure to recommend approval. In disputes between the customer and the auditing team, the tendency became to give the customer the benefit of the doubt and it thus became even easier, in my view, to gain certification to the standard. Even at the time I was doubtful of this approach and now I am convinced of it.

The certification bodies' mistake, in my opinion, was to believe that the service being offered by the certification body was the assessment/audit and that it was that service's value that had to be maximised. It is my belief that the service the certification body offers is the *certification*, and the value that the certificate represents. The audit/assessment is a risk reduction exercise on behalf of the certification body to ensure that companies unworthy of the certificate do not get awarded it, because a certificate that is too easy to obtain is devalued and debased. I believe that this has now happened.

Unfortunately there is a certain inevitability to this, in that one certification body breaking ranks now would rapidly find its business disappear, because a certificate from one body, provided it is accredited, is treated as equal in the marketplace to any other. No one certification body has the ability to differentiate the value of its certificate based on its own service from the certificates of its rivals. Most customers in this environment will take the path of least resistance, although there is still some residual kudos in having approvals from one of the better-known certification bodies than from some of the others. It means that the certification body with the laxest regime threatens to pull all the others down to its level.

The Accreditation Body, which should be the body upholding the standards of certification, in the UK is now profit making in its own right and therefore are anxious to treat the certification bodies as its customers and to give them what they want, which is continued accreditation. Just as the certification bodies are not motivated to seriously challenge their customers' management systems, the accreditation body is not motivated to challenge the certification bodies'.

Implementation by Adopting Organizations

The most unhelpful thing an organization could do, for itself and for the industry, is to

require of its suppliers that they achieve ISO 9000 approval. But this is precisely what many companies did. It has some attractions for a cost-conscious company faced with a concerted review and evaluation of all its suppliers, which for large UK firms could run into thousands. A simple policy appears to address a requirement of the standard which certification bodies have difficulty arguing against, because if they don't believe that ISO 9000 approval gives confidence that the supplier can deliver on quality requirements then who does? However as we have seen it engenders the wrong attitude towards the standard and perpetuates the "certificate on the wall" mentality. Firms will do the minimum necessary to achieve approval and no more. As we have seen from the certification system that minimum is very minimum indeed.

In requirements engineering we are taught that when specifying the requirement for something, it is better to specify a need and not a solution. That is precisely what specifying ISO 9000 as a quality requirement does. The need is quality output from a supplier. One possible solution, and one which may not be appropriate in every case, is an ISO 9000 approval.

Some key requirements of the standard were being widely fudged in many organizations, particularly the monitoring processes of corrective action, preventive action, internal audit and statistical techniques. The certification bodies were if not turning a blind eye to those issues were at least more lax than they had a right to be. Internal audit for example almost everywhere tended to focus on compliance to the organization's own documented procedures and rarely did they offer real improvement opportunities. They tended to behave as "super inspectors", recording as system non-compliances, instances of product failures, usually intermediate products such as unsigned documents or incorrect records. Corrective action often consisted of correcting the mistake, without any attempt to eliminate the causes of the failure. The system remained unchanged after the non-compliance had supposedly been addressed. External auditors, if they spotted recurring problems at all, would tend to raise these issues as observations rather than failure of the system to address a fundamental requirement of the standard.

Inexperienced external auditors themselves would often focus on documentation errors, because they were easy to spot, and miss more important failings relating to the system consistently failing to achieve customer satisfaction. The requirements relating to acting on data at management review were vague in the 1987 standard so external assessors found them difficult to enforce. Management reviews were therefore often superficial and qualitative, but addressing the letter of the standard. It was rare for very senior executives to have more than a passing interest in the management review process, rather leaving that to the quality manager. Seddon [1] points out that, with the standard only requiring one management representative it encouraged senior managers to treat the quality system as almost a parallel system that had little to do with the actual management controls the board exercised to manage the company.

ISO 9000:2000 and supporting Software documents

Criticism of the standard began to mount during the nineties as the revisions of the standards began to take shape. Almost universally the standard was seen as too bureaucratic, too manufacturing-biased, ignoring results, and too open to interpretation.

Often auditors from the same certification body could not agree on the meaning of some clauses and would audit the same company differently on subsequent visits. Although some certification bodies went to great lengths to ensure a consistency of approach the problem was inherent in the language of the standard itself. Critics also pointed out that infrastructures created to win and retain ISO 9000 often were divorced from the actual management of the company. Board level meetings rarely discussed quality, and business plans rarely assessed the impact on the quality system to changes made to the organization or markets. Most quality systems procedures were constantly playing “catch up” to keep it in line with management decisions.

At first the responses to the criticisms were centred around implementation issues, with advocates of differing views arguing about interpretations of what constituted acceptable documented procedures or records, whether signatures were always required or whether electronic records could be acceptable and so on.

The 1994 revision was seen as an intermediate step to more fundamental reforms planned for 2000 and the response was to improve and clarify the language to make it less open to differing interpretations. The changes left few people impressed. Certification bodies played down the significance of the 1994 changes and by doing so missed an opportunity to encourage a change in thinking about the quality system as something integral to business management instead of as a separate world. What was needed was a thorough reappraisal of the philosophy behind the standard to address the criticisms more seriously.

The Quality Principles

ISO 9000 2000 [2] describes the philosophy behind the standard in terms of eight key management principles:

- Customer Focus
- An organization should use its customers’ needs current and future, to shape its policy and processes should be designed to deliver and possibly exceed customer satisfaction and expectations;
- Leadership
- The leaders of the organization should be actively involved in the quality system and should be encouraging the involvement of staff;
- Involvement of People
- All levels of staff should be involved in the development and maintenance of the quality system. Practices should not be imposed from above, but there should be communication and consultation;
- Process Approach
- Procedures should not exist in isolation, but form an interactive element in the overall system and contribute towards the aims of that system;
- System approach to management
- Management should manage the quality system as an integral part of the business, and focus on achievement of process objectives, not just control;
- Continual Improvement
- Is expected to be a permanent objective;
- Factual Approach to Decision making

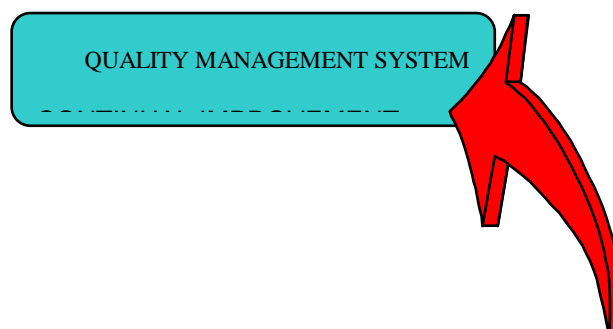
- Decisions should be taken on the basis of analysis of supportable data, and not rely solely on qualitative information;
- Mutually beneficial Supplier Relationships
- Move away from the adversarial contractual position with suppliers but recognises that a helpful relationship which benefits both parties.

From these principles we can already see that, at least in language, the standard has responded to many of the weaknesses of the existing standard as a process improvement tool. The recognition that organizations have inherent interrelating processes to deliver their business objectives echoes the language of the process improvement community at last. The definition of Process in the vocabulary section fits neatly with IDEF0 notation as a transformation of inputs into outputs using resources and governed by constraints.

The Continual Improvement Principle instantly lends further support to the process improvement community. Continual Improvement is defined to be that which increases the probability of enhancing the satisfaction of customers and other interested parties (meaning that a firm is not expected to bankrupt itself to better satisfy the demands of a challenging market!) Actions arising from audit, analysis of customer feedback and review or inspections of product are now no longer restricted to eliminating errors and their causes, current or potential, but expected to evaluate better more efficient ways of working, consuming less resources and so on. This sort of language should strike a chord with senior management who are essentially motivated to do the same thing.

Efficiency is still not explicitly mentioned in ISO 9001, but is mentioned in ISO 9000 and ISO 9004, and so efficiency improvements are a valid pursuit. The ISO 9000 2000 [2] improvement cycle is described in Figure 1.

In the central circle, the four main sections of the standard and their interactions are shown. There is an inner improvement cycle which reflects the old cycle explicit in ISO 9001 1994, [5] of corrective and preventive changes to the system to prevent recurrence of problems and improving the effectiveness of the system elements to deliver defect clear output. The scope for the quality system reflects customer needs, shown by the dashed arrow between the customer and top management. The customer feedback, for which there must now be an established process, arrives as a result of receiving product (or service). As a result of all of these operations, process performance must now be analysed and improvement opportunities taken, reflected by the curved red arrow into the top box.



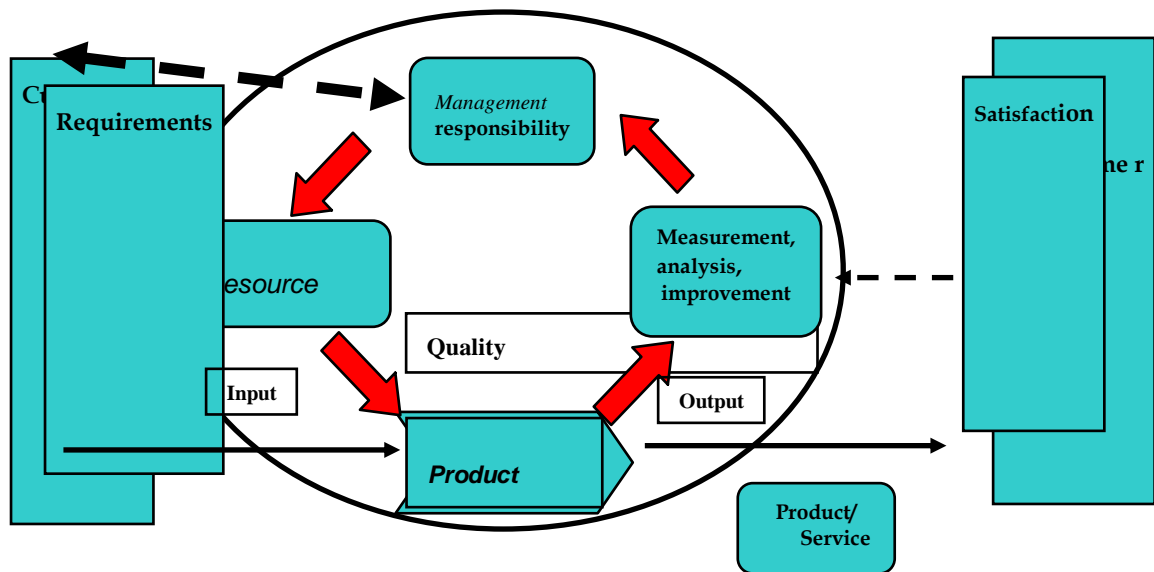


Figure 1. ISO 9000 Continuous Improvement Cycle

The new standard introduces the notion of Top Management, under the leadership principle, defined as the person or group who manage the organization at the highest level, means that there now must be demonstrable involvement in the system from the board level.

Management Reviews must now consider actual data in decisions it takes relating to the quality system. The need to measure process performance is now clearly included, with these measures being fed up to senior management.

The universal criticism that the standard engenders too much bureaucracy has been responded to by a complete change of emphasis on documented procedures. In the generic standard there are only six mandatory documented procedures:

- Document Control
- Control of Records
- Internal Audit
- Control of Non-conforming Product
- Corrective Action
- Preventive Action

It is now incumbent on the adopting organization to determine for itself the extent to which its processes need to be supported by procedures. Processes can now be established in means other than by documentation, which opens up the possibility of using workflow software and automation. The amount of detail required in documentation is and in fact always has been, dependent on the extent of training and qualification of the personnel.

There is now a specific clause in the standard relating to the monitoring and measurement

of processes (clause 8.2.3), which is a mirror of the one for Product (8.2.4). This echoes the guidance in ISO 9000-3 1997, relating to software process metrics. Interestingly, the new draft of ISO 9000-3 [4] suggests process maturity as one aspect of a process that should be measured.

Software supporting procedures

The 1991 and 1997 versions of ISO 9000-3 had already adopted a process approach to the field of software development and its language was already in advance of the generic ISO 9000 standard. As the quality models which pre-date the standard inform the later ones, ISO 9000 has adopted some of the language of the earlier ISO 9000-3, so the software community, at least at the working level, should not experience a great deal of difference in the style of auditing or interpretation of which practices constitute compliance. There may be a difference in management attitudes though, and if by incorporating the key requirements associated with measurement into the standard now, so that they are no longer just guidance, we make some progress on assessing software process performance then it will all have been worth it.

A revision of the TickIT Guide version 5.0 [7] was produced in January of this year to support the provisions of the new standard and Part E has been written to serve as guidance for software until the revised ISO 9000-3 [4] is produced. Part E of the TickIT Guide is completely cross-referenced with ISO/IEC 12207 1995 [6] and it is clearly expected that this will become the governing document for software quality systems.

A working draft of ISO 9000-3 [4] is available at the time of writing and an issued version is due, if not already issued by the time of the conference. If the provisions of the draft survive, the most significant difference is likely to be the encouragement of the use of other software engineering standards such as ISO/IEC 12207, [6] ISO 9126, [8] ISO 15288 and the strongly suggested use of process assessment techniques using ISO/IEC TR 15504, [9] and CMM [10].

It is notable how many clauses of ISO 9000 2000 [2] are left without any extra guidance suggested by the ISO 9000-3 4.1, i.e. 12 out of 28.

Will ISO 9000 2000 deliver Process Improvement?

There is now much stronger support for Process Improvement thinking within the new standard, and process improvement professionals should be able to use it to drive some of the initiatives they need to carry out in order to implement SPI programmes. The stronger links with ISO/IEC TR 15504 [9] and ISO/IEC 12207 [6] are particularly significant in the draft of ISO 9000-3, and of course the TICKIT guide [7] goes further, referring to Bootstrap, Trillium, and CMM [10] as appropriate for process measurement. Given the widespread adoption of the standard in the UK and the growth in Europe this is good news for the process improvement community.

The alternative approach to ISO 9000 that John Seddon [1] advocates is the identification of processes, their process purpose, that processes reflect customer needs and expectations, measuring current process performance and establishing opportunities

for improvement in efficiency by re-engineering the process to eliminate bottlenecks and remove unnecessary steps. These are precisely the measures advocated in the new standard, and so from the ISO 9001 [5] specified requirement point of view, Seddon's [1] criticisms seem to be addressed.

However, the failure of ISO 9000 to deliver process improvements did not rest with flaws in the standard alone. Crucial will be the attitude of the certification bodies, and whether they will tolerate minimal behaviour in some of the new areas and also whether big procurers will persist in their lazy policy of requiring all their suppliers to have approval to the standard.

One certification body is already on record as saying they do not expect to lose customers, in spite of the new standard being more demanding in terms of management involvement and measurement than many companies are currently achieving. If we take this to mean that the audit of ISO 9000 2000 [2] is no more rigorous than currently, then even the new standard will not necessarily improve the situation.

Experiences in Auditing against ISO 9000 2000

It is difficult to overcome a sense of cynicism external audits begin against the new standard as there is a feeling that in the attitude of the certification bodies anyhow, that their approach will be broadly the same. Although experience isn't extensive, as many companies are taking advantage of the three-year transition period, QinetiQ amongst them, some audits have already been done, both against the standard and its previous drafts.

The point of encouragement is that it is now standard policy, in the case of one certification body anyway, to interview the chief executive about how quality objectives have been derived and the CEO's involvement in ensuring the objectives are met.

Auditing compliance is much harder for auditors under the new standard. Where the old process was to check compliance of the documentation with the words in the standard then check the procedures were being adhered to in practice, auditors are now required to evaluate whether the process can be said to be sufficiently established by looking at it in execution only. Whether the process purpose is being achieved is also crucial, which means that auditors now have to look at process outcomes to evaluate whether processes are truly effective.

Many software development companies will still document their processes in the old way. Software companies were always heavily reliant on the quality of their plans, and software auditors always focused on the achievement of plans and their currency. This position will appear not to change, and so software companies will notice less change from the behaviour of their auditors than companies from some other domains.

From my own personal perspective, the wording of the new standard has allowed me, when auditing, to even more strongly encourage the behaviour that I believe is necessary to ensure companies really realise the benefits of an ISO 9000 approval. Then perhaps the advocates of alternative approaches will be able to come together and realise that we are all on the same side.

References

- [1] Seddon J, In Pursuit of Quality, The case against ISO 9000, Oak Tree Press, Dublin, London, UK, 1997.
- [2] ISO 9000:2000 Quality Management Systems – Fundamentals and Vocabulary, BSI, 2000.
- [3] ISO 9001:2000 Quality Management Systems – Requirements, BSI, 2000.
- [4] ISO 9000-3 WD 4.1 Software and System Engineering – Guidelines for the application of ISO 9001 2000 to Software. BSI, 2001.
- [5] ISO 9001 1994 Model for quality assurance in design, development, production, installation and servicing- Formerly BS 5750-1 BSI, 1994.
- [6] ISO/IEC 12207 1995, Information Technology – Software life cycle processes, BSI, 1995
- [7] The TickIT Guide version 5.0, Using ISO 9001:2000 for Software Quality Management System Construction, Certification, and Continual Improvement, DISC TickIT Office, 2001.
- [8] ISO/IEC 9126: 2000 Software Engineering - Product Quality Part 1 Software Quality Characteristics and guidelines for their use BSI, 2000
- [9] ISO/IEC TR 15504 Software Process Assessment (7 parts), BSI 1997
- [10] Paulk et al. Carnegie Mellon University Software Engineering Institute, The Capability Maturity Model. Guidelines for improving the software process. Addison Wesley, 1997.

Comparison of Software Process across Small Companies

Ita Richardson
*Department of Computer Science & Information Systems,
University of Limerick,
Limerick,
Ireland.*

Introduction

The author of this paper developed and verified a technique, the Software Process Matrix (SPM), to help small indigenous software development companies to implement software process improvement strategies [1]. To validate the technique, an implementation phase, using action research with control groups, was designed. Two companies agreed to carry out software process improvement based on the newly developed SPM approach. A further two companies agreed to participate as control companies. This paper discusses software process improvement in these four companies during the longitudinal research which took place over a period of between nine months and two and a half years. Analysis as to why these changes occurred is also presented.

For the purpose of this project, the researcher defined small indigenous software development companies as being within *those companies, with less than 50 employees and less than £3 million turnover, founded in Ireland, who have no parent company, and produce Software products.*

Companies Researched

Four small indigenous software development companies participated in the research

project. It was hoped that the four companies involved in the project would not be involved in any other quality initiatives. In practice, it proved to be very difficult to gain access to four companies who complied with this requirement. All companies researched were based in the mid-west region of Ireland. They were all indigenous companies, some of which had sales and customer service offices overseas, but all of their development was carried out at the parent offices. Their size and the extent of the implementation of Software Process improvement and quality initiatives is summarised in Table 1. To maintain confidentiality, the company names used are pseudonyms.

Company	Total Employees	S/W Dev. Employees	SPI / Quality Initiatives
Computer Craft	16	4	Implemented SPM only
DataNet	9	4	Implemented SPM, also working towards ISO9001 certification
Software Solutions	50	10	Working towards ISO9001.
Ríomhaire	35	3	ISO9001 certified

Table 1
Summary of Companies Researched

In the cases of Computer Craft and DataNet, the researcher was involved in the implementation of the Software Process Matrix within the organisation, using research methods which included participation during meetings, interviewing, observation, self-assessment questionnaires and examination of documentation for two projects within each company. She was involved in discussion about the implementation of these actions and, in some cases, was involved in the writing of procedures to implement the actions. This was the level of participation undertaken by the researcher. She did not become a complete participant in the process, but rather a participant observer, taking on a dual role of “outsider and insider” giving her the opportunity “to participate and to reflect on the data that is gathered during participation” [2].

In Software Solutions and Ríomhaire, the researcher undertook research which involved interviewing, self-assessment questionnaires and examination of documentation for two projects within each company.

The implementations are discussed in more detail in [1] and [3].

Analysis of Changes

During this research project, the author analysed the changes caused by her intervention, but also presents other changes to the software process within all four companies, giving some interesting insights into the software processes of small indigenous software development companies.

Organisation Processes

Ríomhaire had been ISO9001 certified prior to the commencement of the research. DataNet and Software Solutions were both trying to achieve ISO9001 status. Computer Craft were not working towards any formal certification process. At the beginning of the

research project, there had been some element of resistance to ISO9001 in Software Solutions, one of the control companies, but this was not at all evident at the end of the research period. In Ríomhaire, which had been ISO9001 certified for many years, there was an indifference to it within the software development group, who felt that they were not an integral part of the process. This reflects one of the “barriers to acceptance of ISO9001 amongst IT practitioners” as noted by Gillies [4] which is “its generic nature and its origins as a manufacturing standard”, although guidelines for software, ISO9000-3, are now available. There was a particular comment made to the author when she visited Computer Craft - one of the attendees was a former employee of a company who were trying to become ISO9001 compliant, and he remarked that “*the problem (the company) had was that they wrote too detailed procedures, and increased the bureaucracy there*”. Although the software quality assurance engineer in DataNet found that the “*approval process was cumbersome*”, this increase in bureaucracy was not reflected otherwise in any of the three companies who had or were working towards the implementation of ISO9001. In fact, improvement caused by ISO9001 was evident in both Software Solutions and DataNet. This improvement is supported in previous studies. For example, Stelzer et al. [5] found in a study of German companies that “nearly 100% of the company representatives would decide in favour of implementing an ISO9000 quality system once again. They are convinced that the benefits a quality system exceed the costs.”

The implementation of a software process improvement strategy will not succeed without management support. As Weigers [6] concludes, “managers at all levels need to send consistent signals about software process improvement to their constituencies”. In this study, one advantage which all four companies had was that they each had this management support. This was evident in a number of ways. Management were willing to provide the researcher with employees’ time and input to the project. Without any guarantee that their software process would be improved, managers in the action research companies implemented prioritised actions from the Software Process Matrix. All four companies were interested in having the SPICE assessment carried out on their projects and were willing to give the researcher access to documentation to support both the assessment and this research project. In the action research companies, the improvement efforts were also practically supported by the software quality assurance engineer. In DataNet, the role had been specifically created due to the output from the Software Process Matrix and the impetus on the project was maintained by the quality assurance engineer. In Computer Craft, the role was made more specific following the implementation of the SPM actions, giving the quality assurance engineer responsibility for procedures in the company.

Training had become an issue within both action research companies. In DataNet, there was an inability to implement object-oriented design and programming because the knowledge did not exist within the development group. The problem was somewhat different in Computer Craft – the software quality assurance engineer had not been trained in quality assurance, and was trying to learn from the people around her. Nothing had been done within either case, although management in Computer Craft were trying to locate a relevant course. Lack of expertise is one of the characteristics of small companies, and although an action plan existed, there was still a requirement for assistance in the development of procedures. In the course of this project, the researcher was able to fill this role.

On the other hand, within the control companies, training was not seen as an important factor. Both were faced with loss of knowledge as there was natural attrition from the companies, and, in fact, this was noticed in customer support in both organisations. The only impetus towards training was encouraged by ISO9001 requirements, because of which Software Solutions had begun to send its employees on training courses. In each of the four companies researched, training of software developers was not given priority, which is not unusual for employees of small companies - "the emphasis in SMEs in periods of growth is to get the product 'out of the door', rather than to train, on the grounds that this is seen to be taking the workforce away from production activities" [7]. Jones [8] also notes that a "problem with generalist approaches, where the programming or software engineering population carry out development, testing, inspections and maintenance tasks interchangeably" is that "training is usually sparse or lacking for some key activities". Due to the nature of the companies studied, the generalist approach was prevalent. This research supports the views expressed by Jones and Storey.

A difficulty faced by any company, but particularly by a small company is that not only does training cost money, but it can be difficult to release people from projects to do training courses. Another feature observed and worth noting is that no training schedule had been developed for any software developer interviewed. Sanders and Curran [9] in their review of software process improvement consider this a deficiency in any software process - "an important element is an organisation training plan which is reviewed periodically. This plan should identify the skills needed and when they are required". In the changing environment of software, and with much movement of people between companies, each company should seriously consider the development of such training plans.

During the research, two companies, Computer Craft and Software Solutions moved premises. A third, DataNet, had moved just prior to the start of the research. All four companies used open plan layouts, and this contributed positively to the communication within the group.

Customer Management

An aspect to customer management which caused the researcher some concern, particularly in the early stages of the project, was the concept that the software developer's customer was the person to whom they sold their software. In Computer Craft and DataNet, the two action research companies, the customer was not the end user. In DataNet, the customer was seen as either the managing director or the company who would sell the product as a third party product, depending on who was being spoken to. In Computer Craft it was the sales representative. These views are contrary to the views of many quality experts, for example, Juran [10], who describes these as "ultimate users". What is important for the small software development company to remember is that "it is always the customers who judge the quality of our products, whether these are goods or services, hardware or software. Those paying for, or using, the company's products are external customers" [11], and that these customers must be considered when developing product.

By the end of the research period, three of the four companies researched were dealing directly with the 'ultimate user'. The exception was Ríomhaire, but because of the nature of their product, their involvement with customers was naturally different to that of the other three companies. Other employees (hardware engineers on-site) were users of their product, Uimhir, and therefore they had customers available at all times. They also sold Uimhir with the hardware product, and it was normally the Research and Development manager or hardware engineers who met the external customer directly. Customer management did not change in Ríomhaire during the research period.

In the other three companies, the software developers were dealing directly with the customer by the end of the research period, and recognised the benefits of doing so, particularly as it positively affected downstream development activities. In the course of the research project, both action research companies had written procedures to specify and document system requirements and to collect, identify, record and complete new customer requests as part of the actions identified when using the Software Process Matrix, and these actions had contributed directly to the manner in which customers were managed in both Computer Craft and DataNet. In Software Solutions customer management changed because of the influence of two people – the project manager in the client company and the project manager in Software Solutions itself.

Another change which had occurred in the two action research companies and Software Solutions, one of the control companies, was that there had been a rapport built up between the software development companies and the clients. As identified by Keil et al. [12] this rapport is most important when problems arise - "the risk mitigation for customer risks involves relationship management, trust-building and political skills. Project managers must have these skills in order to effectively address the customer risks". When Software Solutions needed to change their project plan due to an employee's illness, the rapport between the project groups was used effectively to avert a crisis within the project.

Only one company, Ríomhaire, carried out competitive analysis. With increasing world-wide competitiveness due to the opening up of the global marketplace with new technologies, it would be wise for the other companies to re-consider their position in relation to this activity.

Engineering Processes

During the research period, due to the intervention of the researcher, both action research companies worked on improving their requirements processes, with the result that this process improved significantly. They had become much closer to the customer, and were able to discuss requirements directly with them. DataNet were developing a package for sale off-the shelf, and their customer was a business partner who had a knowledge of the marketplace. Computer Craft, on the other hand, had a package developed, and were customising their product. Although there was some level of 'feature-creep' in both organisations, the amount had been reduced, making it easier to develop the final product. In Software Solutions, the requirements process also improved, mainly because

the project manager changed the procedures within the company. No improvement occurred in the requirements process in Ríomhaire. There were no requirements engineering techniques used within any of the four companies researched. In fact, it could be stated that writing specifications was seen as covering requirements gathering and no thought was put into the other aspects of the process. Companies researched did no more than elicit requirements, and even then, did not use specific techniques to do so, although the use of requirements engineering techniques should improve the requirements gathering process within the software development companies.

The developers in DataNet had investigated the introduction of object-oriented techniques for both design and programming, but they were not introduced due to the lack of experience and knowledge within the company. No specific design techniques had been used within any of the other companies involved in the research, and like requirements gathering, writing documentation was the 'design process'. While standards for naming conventions and commenting existed in all four companies for coding, there were no conventions for logic, program structure nor program design techniques within any company. While one might be concerned about this, no-one should be surprised, as this has been shown to be the case in other studies. For example Kantz et al. [13] found that of a group of developers trained in a specific methodology, only 17% of them had adopted it in practice. There are, however, some tangible reasons why the companies involved in this research project should consider their use. Previous research has shown that "failure to use effective architecture, design, and development practices is frequently associated with slipped schedules and cost overruns, and is also associated with cancelled projects, although direct management factors are usually more significant in cancellations" and "the conventional structured methods have proven, tangible returns on investment" [8]. Brooks [14] states that "we can get good designs by following good practices instead of poor ones. Good design practices can be taught". Particularly in the case of the small software development company, methodologies should be used with caution as "large projects require much more methodological discipline and formality than small ones" [15].

All four companies had moved toward the use of graphical user interfaces, but only one, Software Solutions, had developed standards for these interfaces. This was encouraged by the project manager. Interestingly, in Computer Craft, they had a product, Data Organiser, which was modified and sold for a U.S. partner, from which they might easily have developed their standards, but they did not do so.

Researchers such as Humphrey [16] propose that "it is wise to conduct an early requirements inspection or walkthrough and to hold a re-review after every major change. To ensure that the requirements provide a reasonable basis for testing, the test group should participate in these reviews". Basili et al. [17] go one step further – "to get high quality software, the various documents associated with software development must be verified and validated". Sanders and Curran [9] establish that a "design defect can be up to fifteen times more expensive to correct at the testing stage than at the design stage". The only one of the four researched companies to use any validation was Computer Craft. Their validation was done at the start of the research period based on the personal experience of the quality assurance engineer, so it was not being done when he was replaced. On the second assessed project, one of their clients reviewed the requirements specification. The introduction of some validation techniques should be considered

within each of the companies.

Testing maintained its prominence as a verification technique, and, although there was a recognition in three of the four companies that code reviews were important, there was no evidence that efforts had been made to ensure that they happened. While code reviews had been included in a procedure within DataNet, they were still not done regularly. Humphrey [16] states that “testing is an inefficient way to find and remove many types of bugs. Software organisations can often improve product quality while at the same time reducing the amount of time they spend on testing.” He agrees that “in spite of its limitations, however, testing is a critical part of the software process”. When testing was carried out, it was very often ad hoc, as defined by Porter et al. [18] where “all reviewers use non-systematic techniques and are assigned the same general responsibilities”. Otherwise, it was based on test plans or checklists. The emphasis on testing was not obvious in the development of test plans in any company except Computer Craft, particularly when projects came under pressure for delivery. In Computer Craft, test plans were the responsibility of the software quality assurance engineer, and detailed test plans were written up and completed for the project studied. The formalisation of the SQA position was as a direct result of the output from the Software Process Matrix. Interestingly, product was sent on alpha test from DataNet without going through any formal testing cycle. Ríomhaire also used customer sites for carrying out beta test, but prior to doing so, they tested their product.

The lack of use of code reviews or software inspections in any of the four companies is also an area of concern. From his extensive work in assessing software companies, Jones [19] concludes that “the measured defect-removal efficiency of inspections is about twice that of most forms of software testing: about 60 percent for inspections versus 30 percent for most kinds of testing.” Humphrey [16] is another advocate of code reviews: “inspections can be highly effective and they should be widely used in software development and maintenance”. He also states that “not only are they more effective than testing for finding many types of problems, but they also find them earlier in the program when the cost of making the corrections is far less”. This researcher proposes that the four software development companies consider code reviews. This is contrary to the view of Jones [8], where he notes that “inspections are seldom used within very small companies with less than ten software personnel, since there may not be a ‘quorum’ for even holding an inspection”. This author believes that an inspection can be held with much fewer people – and that the companies have a lot to gain by including them in their development process.

In Ríomhaire, an ISO9001 procedure was followed to ensure correct and efficient implementation of software within client companies. Within Computer Craft, the installation procedure and build release procedures were written due to the action items output from the SPM. This, combined with improved customer contact at the beginning of the development process, caused improvement of the implementation process. There were no significant changes to implementation in Software Solutions, and, in DataNet, there had been no implementation of the most recent project.

Configuration control has been discussed by Humphrey [16] who considers that “the task of configuration control revolves around one official copy of the code. The simplest way to protect every systems revision is to keep a separate official copy of each revision

level”. Automated code control was used in three of the four companies researched. Ríomhaire, one of the control companies, was the exception to this. Management in Ríomhaire had looked at automated systems previously, and found them too cumbersome for their use. They used procedures to control source code, which worked successfully for them. In the other three companies, code control was proceduralised. Particularly, in Software Solutions, the procedure was written based on current practice, and the process did not change as a result of this.

Project Management

Project management is an important factor in software development companies. In fact, Jones [8] has established that “most successful projects utilize similar patterns of planning, estimating, and quality control technologies” and “deficiencies of the project management function is a fundamental root cause of software disaster”. During the research period, in Computer Craft, DataNet and Software Solutions, project managers had implemented managed and controlled project plans and schedules, and developers were able to identify the tasks that they were working on and were responsible for. In Computer Craft and DataNet, these were implemented as a result of action items from the SPM; in Software Solutions, the implementation was carried out by a project manager who implemented procedures without any external requirement. Project management processes had already existed in Ríomhaire prior to the researcher visiting the company.

Another aspect to project management is risk analysis. The four researched companies dealt with this in different ways. Computer Craft had introduced an element of risk analysis but there was no evidence that this had been taken seriously. Neither DataNet nor Ríomhaire had considered risk analysis at any stage. In Software Solutions risk management had been introduced and used in one project. Sanders and Curran [9] point out that “it is essential for project management to:
identify and assess the risks to a project;
prioritize the risks, and devise plans to reduce them;
monitor the plans and re-evaluate the risks throughout the project”.
This was lacking within the researched companies and should be considered by them.

Project management had not been proceduralised within any of the four companies, and was personalised to suit the project managers. It would be important that, in anticipation of personnel turnover, procedures be written and the process institutionalised, which is achieved “when the process becomes embedded in the day-to-day activities of the organisation” [20].

Support Processes

The support process is an important process, as “the costs and time required to find and fix bugs are the largest contributor to software costs and the most time-consuming portion of software schedules” [8] and should be well-managed to be successful. Within the two control companies, Software Solutions and Ríomhaire, this was indeed the case. Customer support was seen as important within these companies – it was one of their main contacts with customers, and if it did not work well, this reflected badly in the

marketplace. In Computer Craft, the process was managed well externally with the customers, but had not been well-managed internally. Customer support had not arisen in DataNet.

Experiences within three of the four companies were that software, particularly older software, was becoming difficult to maintain. This has been recognised in previous studies, Porter [21] states that “in practice, systems deteriorate and changes become increasingly difficult to implement”. This difficulty was overcome in Ríomhaire by one software engineer taking it upon himself to rewrite code, and proved it to be a successful exercise. The difficulty is that many companies have a lot of code being used currently, and it is only in cases where it needs to be maintained regularly should such an approach be advocated.

Subcontractor Management

While there was some use of subcontractors within three of the four companies, their management was not a prevalent process nor activity within any of the four companies. No significant changes occurred in the management of subcontractors during the research period, nor were there any needs identified which would cause changes to be made.

Software Process Change Management

In considering the effectiveness of the software process improvement initiatives in each of the four researched companies, a framework on change management, presented by Willman [30] shown in Figure 1, was taken into account.

Pressure for change	+	Leadership and vision	+	Capable people	+	Actionable first steps	+	Effective rewards	= Successful implementation
Pressure for change	+	Leadership and vision	+	Capable people	+	Actionable first steps	+	?	= Evaporation
Pressure for change	+	Leadership and vision	+	Capable people	+	?	+	Effective rewards	= Frustration
Pressure for change	+	Leadership and vision	+	?	+	Actionable first steps	+	Effective rewards	= Disengagement
Pressure for change	+	?	+	Capable people	+	Actionable first steps	+	Effective rewards	= Disillusionment
?	+	Leadership and vision	+	Capable people	+	Actionable first steps	+	Effective rewards	= Disinterest

Figure 1
Framework for Change Management

In both action research companies, Computer Craft and DataNet, the successful implementation of actions from the Software Process Matrix caused positive improvement to the software process, particularly to customer management, customer requirements and project management. Considering Willman’s framework, this fits into the “successful implementation” category. What SPM provided in both companies were the *actionable first steps* and *pressure for change* which were combined with the other

factors previously existing in the companies: *leadership and vision, capable people* and *effective rewards*. In DataNet, *pressure for change* had also arisen from the focus to become ISO9001 certified. Positive improvements were also evident within Software Solutions, which also belongs to the “successful implementation” category. In this case, *pressure for change* was there because the company wanted to become ISO9001 certified. The project manager ‘championed’ much of this change, and devised those actions which were implemented within the organisation. In Ríomhaire, there was little evidence of improvement within the organisation, mainly because the company was already ISO9001 certified, and a goal in the current climate was not improving the current process, but rather ensuring that the process would continue to attain this level during audits. The software developers themselves did not feel part of the ISO9001 process and as a group, they did not have any *pressure for change*. The researcher concludes that, for changing the software process, they belong to the “disinterest” category as identified in the framework above. However, in this future, this lack of change in the software process could cause difficulties in Ríomhaire, particularly if customers require a software-based standard such as the Capability Maturity Model or SPICE, as “ISO9001 describes only the minimum criteria for an adequate quality-management system, rather than addressing the entire continuum of process improvement” [23].

Willman also mentions *capable people* as a factor in the success of change management. In the context of the small indigenous software development company, these are the developers on whom the success of the software process is dependant. During her study of the four software development companies, the researcher noted that there was rarely an occasion where a developer was checked as to whether they carried out a process correctly. While this has its faults – processes can become chaotic very quickly – there must also be a trust built between the software development manager and the employee. Otherwise, in such a small group the overhead of checking and re-checking would increase development costs considerably. *Capable people* are indeed fundamental to success within a small company.

Summary

In Software Solutions, one of the control companies, an individual project manager, who recognised that, in order for the company to produce good software, effected change. In Ríomhaire, as the company were already ISO9001 certified, they were only interested in maintaining their current status rather than implementing improvements.

In each of the action research companies, change occurred because the companies used the Software Process Matrix to identify prioritised action items. The most significant changes were in the areas of customer management, collection of customer requirements and project management. These processes were improved because of the emphasis given to them by using SPM, and procedures were written to encapsulate these improved processes. The researcher’s active input was mainly in DataNet, where she became involved in writing procedures in conjunction with the software development group. While it cannot be stated emphatically that the change that has occurred is long-term, procedures which had not previously existed within the action research companies were

written as a result of this research project. There is also a requirement within the companies to use the procedures. Therefore, the effect should be sustainable.

Author

Ita Richardson received her PhD in Computer Science from the University of Limerick in 1999. A lecturer at the University of Limerick, her lecturing responsibilities include Software Process Improvement, Systems Analysis and Design and Computer Graphics. Her PhD research was in the application of Manufacturing Quality Techniques to Software Process Improvement in Small Software Development Companies. She is a founder member of the Small Firms Research Unit at the University of Limerick, whose research is specifically concerned with the growth and development of small firms.

Companies

Computer Craft Ltd. was established in April, 1984 to develop software for use in specific business functions. The company, based in the mid-west region of Ireland, employs 16 people in the Irish office and in a sales office in the U.K. Of these, the software development group of four people is based in Ireland. They have a strategic partnership with a U.S. software company.

DataNet is an indigenous small software development company located in the mid-west Region of Ireland. Founded in 1996, the company are engaged in developing hardware and also in software for the electronics sector and other business functions.

Software Solutions was founded in Ireland over ten years ago by the current Director/Manager. They supply machinery to perform a specific industry function. These machines are supported by application software, which is developed at their Irish office. There is also a UK Sales company in the group.

Ríomhaire is an electronics company which employs thirty-five people. Their manufacturing plant in Ireland also has a Research and Development group. The software group, consisting of three software engineers, is part of this Research and Development group although other engineers sometimes get involved in software development. Ríomhaire also has sales offices in some European companies and in the United States of America.

References

- [1] Richardson, Ita, 1999. *Improving the Software Process in Small Indigenous Software Development Companies using a model based on Quality Function Deployment*. PhD Thesis, University of Limerick.
- [2] Burgess, R.G., 1982. Some Role Problems in Field Research in *Field Research: a Sourcebook and Field Manual*. Robert G. Burgess, editor, George Allen & Unwin (Publishers) Ltd., London, pp 32-45.
- [3] Richardson, Ita and Kevin Ryan, 2001. Software Process Improvements in a Very Small Company. *Software Quality Professional*, Volume 3, Issue 2, pp 23-35.
- [4] Gillies, Alan C., 1992. *Software Quality, Theory and Management*, Chapman & Hall, U.K.
- [5] Stelzer, Dirk, Werner Mellis and Georg Herzwurm, 1996. Software Process Improvement via ISO9000? Results of Two Surveys among European Software Houses. *Software Process – Improvement and Practice*, Volume 2, Issue 3, September, pp 197-210.
- [6] Weigers, Karl, 1996. “Software Process Improvement: 10 Traps to Avoid”, *Software Development*, May, pp 51-58.
- [7] Storey D. J., 1994. *Understanding the Small Business Sector*, Routledge, London, U.K.
- [8] Jones, Capers, 1996. *Patterns of Software Systems Failure and Success*, International Thompson Computer Press, Boston, U.S.A.
- [9] Sanders, Joc and Eugene Curran, 1994. *Software Quality A Framework for Success in Software Development and Support*, Addison-Wesley, U.K.
- [10] Juran, J.M., 1988. *Juran on Planning for Quality*, The Free Press, New York.
- [11] Bergman, Bo and Bengt Klefsjo, 1994. *Quality from Customer Needs to Customer Satisfaction*, Studentlitteratur, Sweden.
- [12] Keil, Mark, Paul E. Cule, Kalle Lyytinen and Roy C. Schmidt, 1998. A Framework for Identifying Software Project Risks. *Communications of the ACM*, Volume 41, No. 11, November, pp 76-83.
- [13] Kantz, Karlheinz, Jan Pries-Heje and Lone Malmberg, 1998. Is it all in vain? The effect of Systems Development Education on Practice: The Multiview example in Wood-Harper, A.T., Nimal Jayaratna and J.R.G. Wood, editors, *Methodologies for Developing and Managing Emerging Technology Based Information Systems, Proceedings of Sixth International Conference on Information Systems Methodologies*, Salford, 25-27th August, Springer-Verlag, London, pp 230-247.

- [14] Brooks Jr, Fredrik P., 1986. No Silver Bullet – Essence and Accidents of Software Engineering. *Information Processing*, pp 1069-1076.
- [15] Glass, Robert L., 1996. Through A Glass, Darkly. *DATA BASE Advances*, Volume 27, No. 1, pp 14-15.
- [16] Humphrey, Watts S., 1989. *Managing the Software Process*. Addison-Wesley, Reading, M.A., U.S.A.
- [17] Basili, Victor R., Scott Green, Oliver Laitenberger, Filippo Lanubile, Forrest Shull, Sivert Sorumgard, Martin V. Zelkowitz, 1996. The Empirical Investigation of Perspective-Based Reading. *Empirical Software Engineering, An International Journal*, Volume 1, No. 2, pp 133-164.
- [18] Porter, Adam A., Lawrence G. Votta, Jr., Basili, Victor R., 1995. Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. *IEEE Transactions on Software Engineering*, Volume 21, No. 6, June, pp 563-575.
- [19] Jones, Capers, 1996. Our Worst Current Development Practices. *IEEE Software*, March, pp 102-104.
- [20] Zahran, Sami, 1998. *Software Process Improvement, Practical Guidelines for Business Success*. Addison-Wesley, U.K.
- [21] Porter, Adam A., 1997. Fundamental Laws and Assumptions of Software Maintenance. *Empirical Software Engineering*, Volume 2, Number 2, 1997, pp 119-131.
- [22] Willman, Paul, 1996. *Organisational Change in the 1990s*. Lecture at University of Limerick, 10th October.
- [23] Paulk, Mark C., 1995. “How ISO9001 Compares with the CMM”, IEEE Software, January, pp 74-83.

Session 7 - SPI and Virtual Organisations

**Session Chair:
Eugene O'Leary, Tecnet, Ireland**

MEDIA-ISF Information sans Frontieres	7-2
Experience with Team-work Concepts for Virtual Quality Management	7-20
SPI repository for small software organisations	7-42

MEDIA-ISF Information sans Frontieres

Dr. György Fóris
BruxInFo, Brussels, B

János Ivanyos
MemoLuX, Budapest, H

Dr. Richard Messnarz
ISCN, Bray, IRL

Abstract.

MEDIA-ISF is a best practice action under the 5th framework of the European Union. It builds towards a new and innovative model for e-working, e-commerce and dissemination in Europe building a bridge between relevant information on best practice gathered in European regions and Europe's media (newsagencies, journalists, newspapers, portals, and so forth).

The MEDIA-ISF model is building from existing blocks of former European technology, methodology and business results (such as NQA Integrated Teamwork and Network based Quality Assurance model and tool, EFQM Excellence Model, BESTREGIT innovation transfer experiments, multi-media based information services, Internet technology). The novelty of the model comes from the unique combination of different type of services customised for different sectors and regions in the same structure, enhancing business and communication co-operation between member states and candidate countries, showcasing the benefits and facilitating their deployment in SMEs. The first implementation of the principal information services is organised around the specific digital contents of “Media on EU Enlargement” and “Virtual Organisation”. Further implementations will be based on the selection of new contents to extend the model into other business sectors (e.g. software engineering, employer services, public sector, etc.) by interconnecting the regional and sectoral communities in a multilingual network environment.

The Need for Such a Model

It is stated by “The European Observatory for SMEs“ (July 2000, submitted by the EC) that the barriers of the expansion of e-commerce for SMEs in Europe “derive from different sources, namely the characteristics of SMEs themselves, the consumers, technology and the legal framework”. It is also mentioned, that the “lack of information and lack of awareness of good examples concerning e-commerce, may cause the perception that selling products or services on the Internet does not apply to the enterprise” and “ the three most important barriers following after this one are: doubts about return on investment, lack of skilled personnel and lack of consumer access to the electronic market”. Also realised that “language is still a prevalent barrier”, because “the Internet, primarily being an English speaking arena, is therefore largely out of reach for many potential customers with another native tongue”. Besides of the high telecom prices and uncertainty about Intellectual Property Rights, which can slow down the supply of content and take-up of new services of the network economy, the primary obstacles of exploiting e-commerce in Europe are identified by the industry experts, such as:

- Under-utilisation of public sector information in Europe.
- Potential of cultural and linguistic customisation to reduce trading barriers between SMEs and their potential customers.
- Insufficient investment and market transparency in Europe, especially for SMEs.

The targeted SMEs independently their regional and sectoral (media or technology) status are suffering from the above mentioned barriers acting as brakes to developments and opportunities for content producers in Europe. The media sector is under the pressure of the commercial and often the local governmental indications, the public sector information providers have no real direct presence in the marketplace, and the linguistic customisation of the materials by the local public service companies are not efficient enough to reach the whole society. On the other hand small business is not represented well in the profit earning marketing channels, nevertheless the significant part of the European media content is produced directly by them.

The whole digital content industry is in fast expansion. The new mobile Internet enabled technology is creating an additional demand of content to be tailored to these new device types. The present status of distinct circuits of production and distribution of different types of content (publishing, audio-visual, etc.) is changing due to the convergence of digital technologies. Established content providers are being challenged from previously unrelated industries, like telecom and software companies, and the new mergers and acquisitions of large media firms are combining the ownership of content and the control of the distribution channels. The large firms are operating more and more at a global level, which opens new opportunities for small firms to find market niches in the terms of geographical location and specialised product.

In order to answer these challenges, the MEDIA-ISF project establishes a “catalyst” information service platform. By this “catalyst” role an implemented virtual organisation structure will help distributing best practice information to SMEs across Europe by a virtual organisation connecting European media, journalists, SMEs in different regions of Europe.

Overview of the Service Model

The MEDIA-ISF model is focused on the “catalyst“ function provided by the interconnection of different digital content groups. The connection of the different content groups is managed by a Virtual Organisation. A 3-level structure for information services will be established for each content group. This service architecture provides a master model for establishing cost-effective and high impact interconnections between the SME user organisations and European media within the global digital economy.

MEDIA-ISF is established on the following novel 3 level architecture of information service delivery:

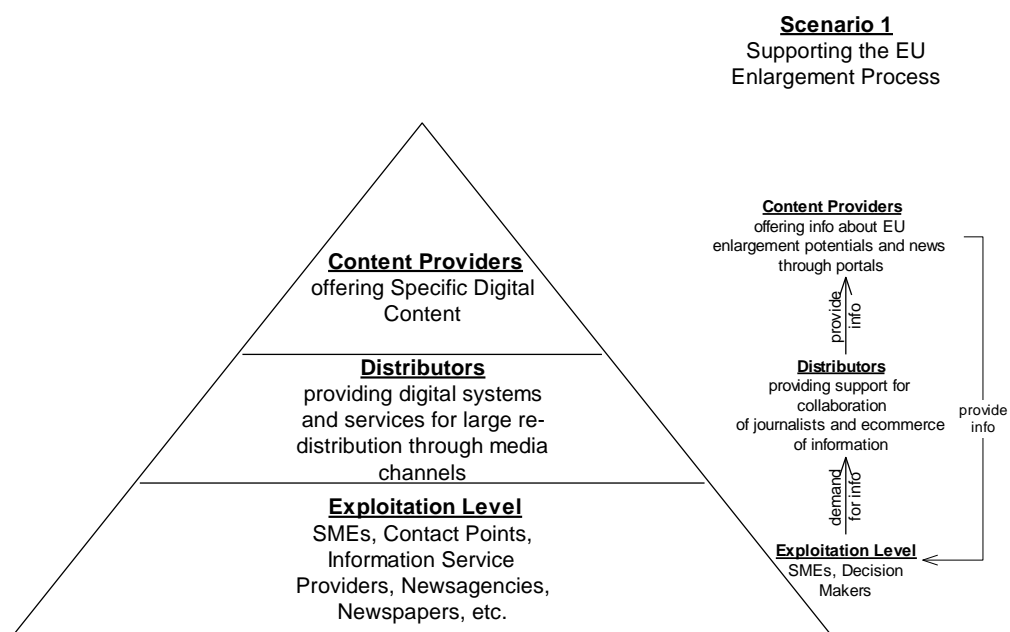


Fig. MEDIA-ISF.1: A three - level service platform architecture for large scale information distribution

- The first level, called “Content” is the generic information source regarding specific digital content, in this case this is about the model itself, the methods, the tools, etc. selected and used in the organisation. The maintenance of this information source and the “e-work” server providing collaborative working tools is performed by the jointly approved and appointed group member(s).
- The second level, called “Distribution” consists of SME business administration service provider(s) and makes the first level information available for the registered users, but also provides the “e-commerce” platform, manages the transactions through the network, does the local customisations, and performs all the administrative tasks regarding the business models available via the network

organisation.

- The third level, called “Exploitation” consists of the SME contact points providing general training and marketing services regarding exploitation activities, keeping contact with the general portal service providers and the non-registered users. Each of the other content groups of the virtual organisation shall have a nominated contact point representing its content specialities at this level of the “Virtual Organisation” content group.

The Virtual Organisation Concept (VOC)

The virtual organisation is established in such a way that it can be adapted to different content groups in Europe. In a first trial the EU-Enlargement topic will be implemented with the following architecture.

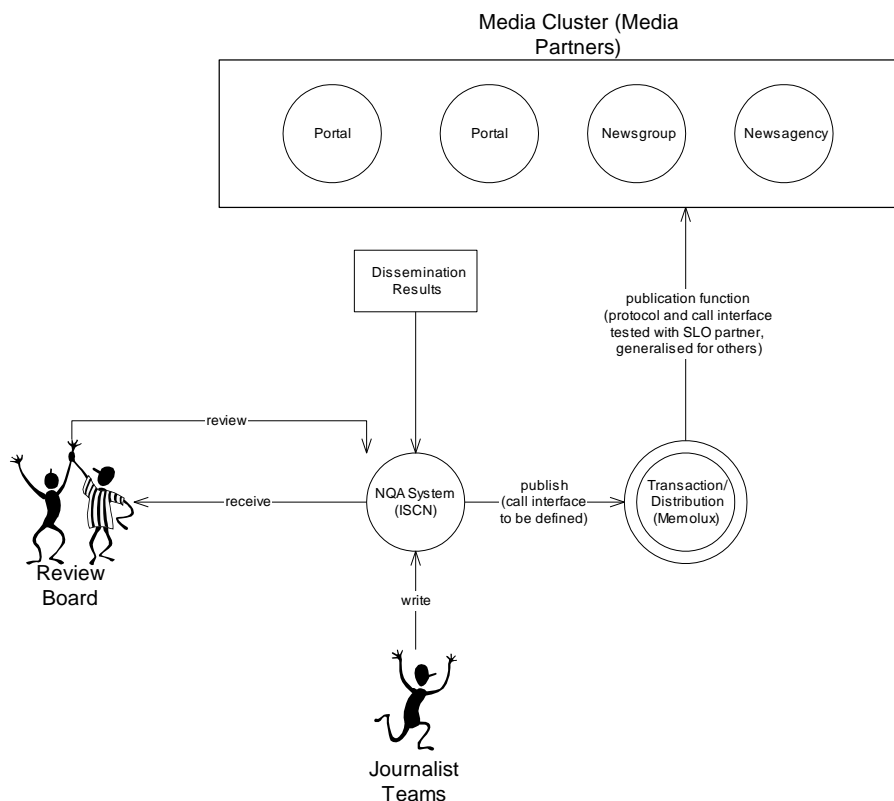


Fig. MEDIA-ISF.2: The virtual organisation concept (VOC) implementing the 3 - level service architecture for the EU-enlargement topic

The working scenario

As a best practice action, the journalists across Europe are working together through an NQA server configured with a team-work scenario for collaborative authoring of information concerning EU enlargement. A review board of representatives from journalists and news agencies will review these contributions through the virtual office. Information getting the status accepted can be published.

By choosing PUBLISH on the result of a teamwork through the NQA server, this article is transmitted to the transaction system which automatically recognises the language, determines related portals, news groups, and news agencies and through a call interface and agreed protocol publishes this information through media across Europe.

The virtual organisation is based on existing information technology solutions, which have been developed in previous EU research and technology programmes:

- NQA (Teamwork and Quality Assurance over the Internet)
- BESTREGIT (BEST REGIONal Technology Transfer)
- eTRAP electronic transaction processing solution (a highly adaptable and large scale transaction software system) based on former e-commerce practices
- Portals and Information Services (EurActiv, BruxInfo, etc.)

NQA (Network based Quality Assurance)

The idea of this initiative goes back into the beginning 90s when big firms in automotive industry realised that in supply chains they work together in kind of virtual organisations. You could not draw exactly the border between firms, and teams were interdisciplinary with members from different firms working together on sophisticated technical solutions. At the same time Eastern Europe opened its borders and methodologies and technologies were discussed to establish a working platform for East-West development collaborations which allow quality management through the Internet. This led to the fact that an NQA (Network based Quality Assurance) platform was supported by a number of semi-public innovation centres putting it in place as a collaboration platform solution from 1998 onwards.

The NQA approach bases on three principles which have been discussed and published at previous ISCN conferences (<http://www.iscn.ie/conferences>) and about which a book has been published by IEEE [3]: Better Software Practice for Business Benefit - Principles and Experience (ed. Richard Messnarz, ISCN).

The three principles:

- Role and information flow based team work process management (role and people - centred teamwork)
- Development by configuration (Data and functions are adapted by configuration)
- Re-Use pool concept (Re-Use Pool of existing functions)

A major feature to make such a virtual approach applicable for different environments is that such a system must be kept completely configurable. The menu, the data, the functions, the document/information flows can be configured for different user scenarios and this high configurability is the major feature of an NQA virtual office.

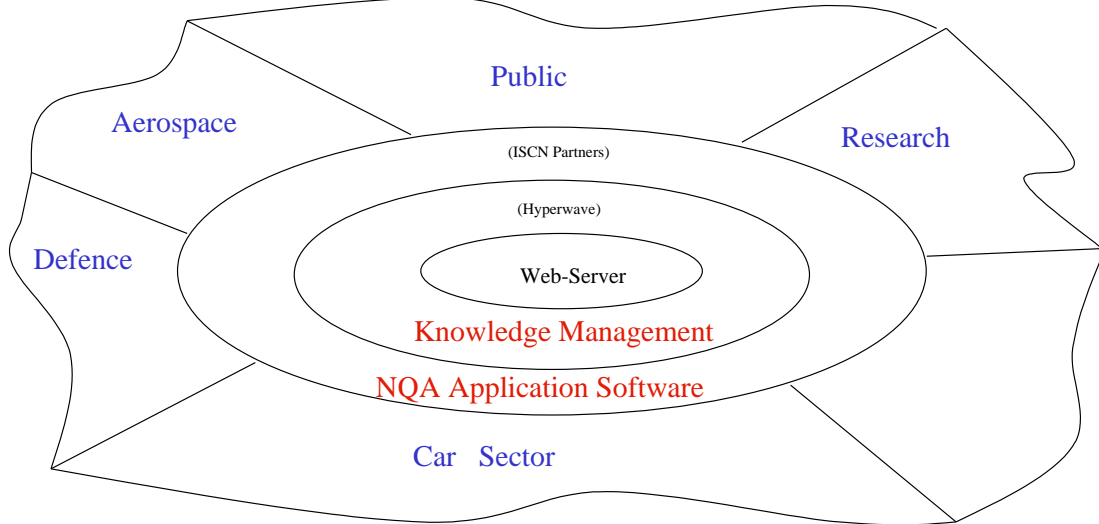


Fig. MEDIA-ISF.3: The virtual organisation implementing the 3 - level service architecture for the EU-enlargement topic

More information is available regarding NQA experiences at: http://www.iscn.at/select_newspaper/qa-systems/nqa.html

Transaction Processing System (eTRAP)

Memolux, the Hungarian Information Service Distributor, provides its running application hosting infrastructure and its service oriented quality system documentation and workflows (developed during the PASS ESSI PIE project and have been re-worked based on the requirements of the new ISO9001:2000 quality standard). The information service distribution platform of the Memolux e-commerce activities has been implemented in the way appropriate to the MEDIA-ISF virtual organisation concept. See also an article at: <http://www3.interscience.wiley.com/cgi-bin/abstract/76503384/START>

The eTRAP solution takes a key part in Virtual Organisation Concept. This system enables the distribution of contents among the exploitation level members. The technical modelling of the VOC is that exploitation level generates demand for content through querying the eTRAP. eTRAP therefore will provide appropriate contents. This assumes that contents are submitted by the content provider organisations.

An exploitation organisation (i.e. a portal) can be utilised differently. Less content will take us to handle more query and vica versa. We assume the portal utilization shall be formulised. The portal utilization at the eTRAP side is $U_{eTRAP} = Q_{eTRAP} \cdot C_{eTRAP}$, where Q stands for number of queries (in a given time period) transmitted to eTRAP for requesting contents, C stands for the amount of content provided previously to the portal. U means, the utilization of the eTRAP interface facing the given portal.

eTRAP shall be connected to significant number of portals, and other sites. Concerning the utilization of interface (remember, Q must be transferred to eTRAP, and results shall be transmitted back to portal) we need decrease the number of queries arriving from the portal.

The users – the consumers of the exploitation level organisations – are keen on response delays. This leads us to a concept where we try to minimize the delay of a query. On the other hand the publishing procedure duration might increase.

The organisation roles are mixed. A content provider may play an exploitation level role as the circumstances are dictating it. Therefore we shall be able to provide organisation with content level and exploitation level services, sometimes dynamically.

According with the premises described above, we can easily understand why eTRAP concept looks like this:

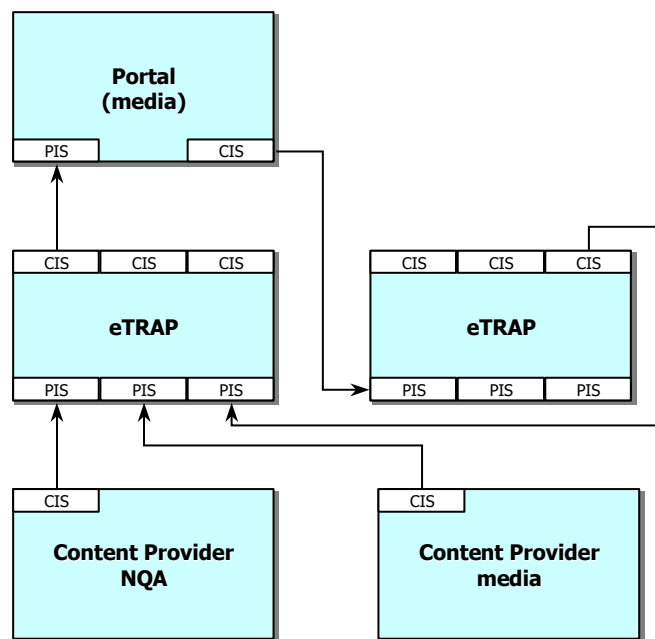


Fig. MEDIA-ISF.4: The eTRAP concept

There are two significantly different interfaces mentioned above. CIS – Content Information Subsystem is a module. CIS enables the content transfers on demand. A PIS

– which is a portal information subsystem – sends requests to CIS, and caches the provided contents. The figure also illustrates how a portal (having a PIS) can play a content provider role (having CIS).

The configuration enables having multiple eTRAPs in the same system. Multiple VOC might be formed in the evolution phase, which will have the ability of integrate their separated VOC services – simply creating PIS/CIS interfaces to each other.

The arrows – which are representing the transfer of contents – might form a loop. A loop means here that a content transfer initiated by an eTRAP returned back. To avoid the infinite looping of content – and queries, which might occur infinite response time – we need to design eTRAP to find loops and stop avalanching.

eTRAP contains CIS and PIS interfaces, as a PIS always talks to a CIS and vice versa. eTRAP has multiple PIS and CIS interfaces, therefore an internal bus system is designed to enable the sequential, or parallel processing of queries and content submissions. The following figure represents the internal modularity of an eTRAP:

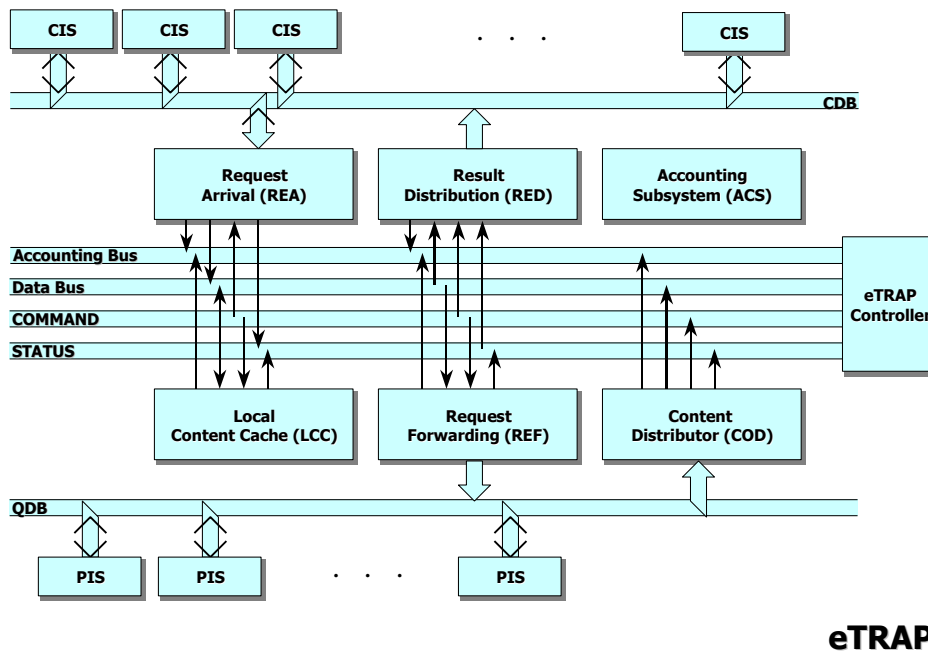


Fig. MEDIA-ISF.5: eTRAP internal modularity

eTRAP contains seven modules for different purposes and any number of CIS, PIS interfaces to communicate with other servers. The modules' purposes are as follows:

REA: Responsible for the arrival process of a request (from a portal). Identification, and acknowledgments are sent by REA backwards.

RED: When a request has been processed the result (say the content) shall be transferred to the query source. RED transfers the content through one, or more CIS interfaces.

LCC: Local Content Cache is managing the published contents. It caches the contents and enables to query the contents.

REF: Request forwarding. If the eTRAP configuration is set to do so, requests might be forwarded to other content provider or distribution systems (i.e. another eTRAP, or another content provider).

COD: Content distributor module enables the distribution of contents among the exploitation level organizations (through caching and notification services).

ACS: Accounting Subsystems provides audit trails for performance analysis and business purposes.

Generalisation of the Concept

The overall service architecture and the technical concept discussed in the previous chapters of this paper can be adapted "by configuration" to many different content groups. An adaptation would just require

- the configuration of new collaborative scenarios on NQA,
- the adaptation of parameters to be given to the call interface to the transaction system and from there to the European portals

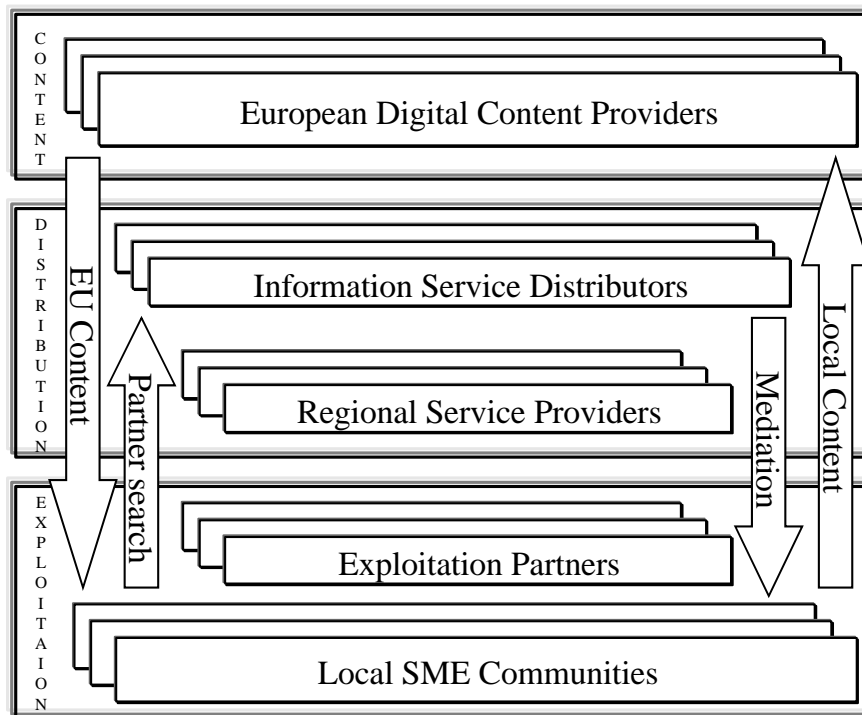


Fig. MEDIA-ISF.6: The MEDIA-ISF Exploitation Model

The European level Digital Content Providers producing the “Content” level information gain more publicity and contributions through a European virtual organisation providing and generating content through distributed authoring teams.

The Information Service Distributors and Regional Service Providers of the

“Distribution” level still have their existing business connections, technology and methodology background to provide information services, such as news-making, consulting, out-sourcing, training, etc. They have the business benefit from connecting their clientele together via the virtual network organisation providing much more information services for their own clients than traditionally. The Distributors set up the client groups of the virtual organisation based on the local knowledge of business needs and environmental issues like local legislation requirements and connection to existing regional services. They - together with the connected Regional Service Providers - have the rights to transform the information and knowledge prepared by the “Content” level to the regional market, giving added value by the knowledge of the audience environment (translation, customisation, localisation, delivery, etc.). Their position - being a kind of platform for information-transfer and mediation between local end-users and other regional providers (or the Content level provider itself) - offers the opportunity for them to enlarge the scope of their information and business network. Actually their business benefit is also comes from the income of the additional services towards their existing clients and performing requests the additional clientele provided by the virtual network organisation. As key stakeholders of the organisations, they initiate to establish new information services through the network organisation utilising the process oriented knowledge and learning management supported by information service scenarios of the virtual organisation.

The Exploitation Partners contribute to the “Exploitation” level by their existing training and knowledge sharing activities utilising the additional market positions. The typical interest of participating at “Exploitation” level of the model is to improve sales and marketing channels towards client-networks. Bestregit technology transfer experiment is adapted to the needs of Exploitation Partner organisations, used as a methodology for business and organisational goal and workflow definition describing the commercial and technical features of the “Exploitation” level participation.

The end-user SME communities that are connected to the “Exploitation” level of the virtual organisation will have the business opportunity to participate as active partners in the information flow.

A Future Business Scenario for Europe

In a long term future plan the end users can apply the information services for not only to satisfy their own needs, but to participate within other information service procedures as providers for the much larger virtual market. Using the new information service applications hosted by the Information Service Distributors, they can gain additional market and can utilise their own abilities and enables in a competitive way by building interactive connections with other participants directly, regardless at which level they have their activity.

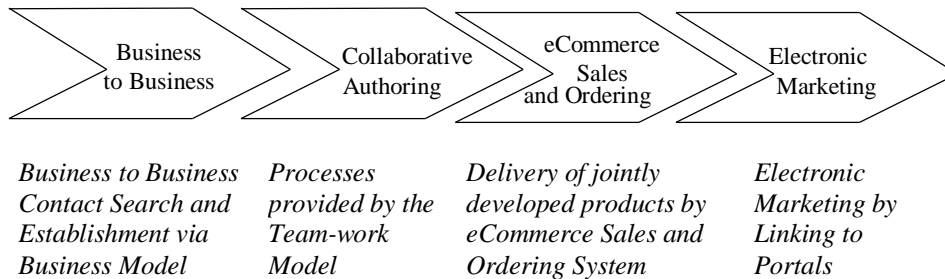


Fig. MEDIA-ISF.7: The Future Business Scenario

Based on the results from MEDIA-ISF a cluster of NQA servers and transaction systems connected with European portals and media could be established forming a service platform for a set of European digital content topics.

Equally important benefit can be for SMEs to get through the transaction system an easily available mediation and translation service which can help them to cross language borders as well. Following the processes of the organisation management information service, the small business partners and individuals can fulfil special quality requirements of bigger consumers and can involve other participants from the organisation as resources in a managed and visible way.

Transforming media SME into smart organisation

By participating in the MEDIA-ISF Best Practice Action at the “Distribution” level, BruxInFo validates the business sustainability of converting traditional journalism into virtual service by the support of the 3 level model architecture. The long term business aim of the BruxInFo is to create and run a comprehensive, well structured, multimedia-based interactive Internet service with well defined EU-content, which is - at the same time - an organic part of a European wide network with similar structure and content service. The BruxInFo service is to be based on three input levels: 1. Materials from the EU level content service (EurActiv), 2. The activity of the Brussels based staff-members of BruxInFo, 3. The activity of its Budapest based staff-members. Within this structure BruxInFo will automatically transfer the first level (English written) material to its registered clients on the one hand, and will establish its Hungarian language service - with parallel structure of that of the EurActiv's one - on the other. Within its Hungarian service it will localise (translate, select) the first level material, and will add locally collected/edited inputs (still based on EU-oriented content, but with special Hungarian angle or origin).

The final outcome is supposed to be available for identified subscribers (regional service providers and their environment in Hungary). The technological vehicle for this is an interactive network - still under the business control of BruxInFo - which enables its users to establish ad hoc or regular virtual editorial offices among themselves (or with members of other similar networks which part of the European wide MEDIA-ISF model). They all are eligible and capable to contact to each other, or to other second level distributors, (or to their third level clients), likewise to the first level provider, too - in order to offer or demand materials, or mediate among different participants. One of the crucial roles of the BruxInFo in this respect to offer mediation (partner search, mediation

and translation if it is necessary) for those who need it. The targeted BruxInFo service is supposed to have powerful searching engine(s) and a permanently broadening database as well.

The unique values of the foreseen service are rooted in its three level structure, in its interactivity, and in the fact that within the framework of the MEDIA-ISF model several similar ("BruxInFo-kind") structures exist in parallel, among which the same interactive co-operation and information exchange is feasible and even encouraged.

In parallel of implementing the media content business case, two additional directions are gradually explored:

Enlarging the geographical scope of the media content

After the model having been established and fully operational, the service providers of the "Media on EU Enlargement" content-group - indicate interest in enlarging the geographical scope of the media model. The strategic aim is to enlarge both the services and the subscriber-base in order to make the model more and more attractive, leading a self-generating circle of growing number users and growing benefits.

Enlarging the contents of the MEDIA-ISF with new models dealing with other activities

Utilising the established dissemination system the MEDIA-ISF framework will be adapted to further fields (as payroll-service, or different kind of IST-service e.g.), using the same structure of the model. By this step the strategic aim is also broadened: to get enlarged services both in terms of geographical (cross-regional) and cross-professional scopes, further more generating all kind of interest to join the model and to profit from its benefit.

Success measures of business objectives

Progress and success metrics related to the MEDIA-ISF business objectives are compared with a baseline (current practice), short term and expected (for the second year from the start) status:

	Baseline	Best Practice (short term)	Exploitation (expectation)
Increasing customised volume and topic of information available for network participants	Existing separate portal services of EurActiv, EON, Bestregit, IRCA, etc.	New MEDIA-ISF Portal linked to the extended EurActiv, EON, BruxInFo, APS portals	Dozens of interconnected portals
Increasing visibility (hits)	100.000 hits/month	Over 500.000 hits/month	Over 1.000.000 hits/month
Increasing number of registered end-users	N/A	Up to 2000	Over 4000
Increasing volume of activity types available for registered media SMEs	Searching	Searching and Teamworking	Commercialising
Increasing participation in the learning processes	EJC Trainings, Bestregit	Additional Training Activities and new modules for existing courses	Additional self-learning by the MEDIA-ISF Model
Increasing number of active end-users providing service by MEDIA-ISF	N/A	Min. 10	Up to 200
Increasing volume of end-user requested added value	None	10 article/month	Up to 200 articles / month
Increasing user satisfaction	Implementing rating mechanism	Presenting the metrics	Follow-up the measured satisfaction data
Increasing productivity	Cutting the average cost/web article	Cutting the average cost/customised web article	Cutting the average cost/commercialised content page
Increasing volume of new partnership/cooperation in the value chain	Consortium for MEDIA-ISF project	MEDIA-ISF Best Practice Partnership (2 content groups)	5-10 new content groups within 2 years
Increasing volume of interconnections of different business groups based on multi-linguality and multi-regionalism	MEDIA-ISF project	Interconnection of the media and innovation business groups of UK, IRL, B, NL, A, SIT, H	Extension to Enlargement countries and other European regions
Higher Quality Control on virtual collaboration	Non standard methods	ISO9001:2000 conform workflows	Excellence Model
Increasing volume of technology transfer	Bestregit	NQA technology implemented for MEDIA-ISF	Information Service Distribution for NQA and other new tools
Process Improvement	Bestregit	Media authoring & distributing	New content related processes
Increasing role in policy and strategy making of the MEDIA-ISF organisation	MEDIA-ISF proposal	MEDIA-ISF Exploitation Plan	Extension towards other Exploitation Models

Table. MEDIA-ISF.8: The MEDIA-ISF Objectives

Results and Lessons Learned

The implementation of the MEDIA-ISF model directly contributes to the following aspects of economic development:

- Improving the European level digital content collection, distribution and usage in order to stimulate economic activity on global networks especially for SMEs
- Fostering new partnerships and the adoption of multilingual and multicultural strategies in order to facilitate the economic and social integration of nationals of the applicant countries into the information society
- Improving awareness of and exposure to new methods, tools and processes in order to create favourable conditions for the reduction of market fragmentation in terms of marketing, distribution and use of European digital content potential

The business efficiency of the MEDIA-ISF model depends on the following matters:

- critical mass of participants,
- growing number of services involved in the model,
- opened and flexible structure enabling all the participants to get different, easily variable roles within the framework of the model according to their interest and ability,
- autonomy and self-accounting for all the participants - Exploitation Partners, Regional Service Providers Information Service Distributors - within the scope of their activity.

MEDIA-ISF is trying to create a new structure for dissemination of EU results on a large scale. It is not creating, like in many traditional EU dissemination projects, a consortium and set of regional workshops across Europe. It is utilising the existing media of Europe to reach the regions of Europe via newspapers, news agencies, portals of existing information providers and connecting them through a virtual editing and transaction system.

Authors and Companies

Dr Gyorgy Foris

Manager of the BruXInFo, cofounder of the PecOffice, office and working community of Brussels based Central-European journalists, established in 1998. Graduated as Economist at the Budapest University of Economics in 1979, defended his PhD on International Relationships in 1989. Received diploma of the High School of Journalism of Aarhus and the High School of Journalism of Utrecht (1990-91 academic year) completing the "European Year" course sponsored by the European Commission. Stager in the European Commission in 1991 (four months at DG X.). Between 1992-97 resident correspondent of the Hungarian News Agency in Brussels, covering EU-related events. Since 1997 the Co-editor and author of the EuroAtlantic Newsletter, twice a weekly bulletin published by the Hungarian News Agency and focusing on EU issues. Between 1997-2000 resident correspondent in Brussels of the "Magyar Nemzet". Since 2000 permanent Brussels based correspondent of the "Vilaggazdasag", biggest Hungarian national economic daily. Regular Course Leader and lecturer of the European Journalism Centre.

Mr. Janos Ivanyos

Mr. Janos Ivanyos is one of the founders of company Memolux and the managing director responsible for Information and Communication Technology support and development since 1989. He was graduated as an economist at University of Economics, Budapest in 1984. He was working for the Computer Center of the National Planning Office for 4 years and then established his first computer service company. He managed several IT projects including IT outsourcing services to Unilever Hungary, Heating Works of Budapest, National Employment Office, Ministry of Finance, etc. He was the Project Manager and Contractor for the PASS Esprit project during the period of 1997-1999.

Dr Richard Messnarz

Dr Richard Messnarz made his MSc at the University of Technology in Graz in Austria. In 1995 he received a PhD from the same university for the work on a QUES - "A Quantitative Quality Evaluation System". He has participated in a number of European research and industry projects since 1990. He was also the project leader of the Comett expert exchange project ISCN in 1993 which led to the foundation of ISCN itself in 1994.

He is the Executive Director of ISCN, the technical co-ordinator of the PICO (Process Improvement Combined approach) initiative, the co-ordinator of the software engineering group within the pilot project for building a prototype of a virtual university, and editor of a book published in 1999 from IEEE "Better Software Practice for Business Benefit - Principles and Experience". He is a chairman and main organiser of the EuroSPI conference series (the former ISCN series).

BruxInFo s.p.r.l. (B)

The BruxInFo was registered in Belgium, in year 2000. Its main profile: media-service, media training and programme-management, information service and consultant, covering first of all EU-integration related subjects and developments. Its staff members are Brussels based professional journalists, with profound experience of EU-journalism. The aimed business benefit for the BruxInFo is to create an overall and comprehensive, EU-related, Internet based information network in Hungary, with a strong background in Brussels on the one hand, and with the possibility to establish virtual editorial network among different regions or countries on the other hand, targeting at Hungarian media SMEs and other interested potential consumers. Its service is to focus on the enlargement process in short term, extending the covered subject field to the SME related economic news and regional policy in longer term.

Memolux (HU)

Memolux, established in 1989, is a Hungarian private SME company with professional experience as a service provider in finance and public accountancy, management organization, software development and information system engineering. In Hungary, Memolux is ranked after the "Big Five", the five greater advisory firms in public accountancy. Memolux is a member of several economic chambers (AMCHAM, BCHH, CCCH). The payroll and accounting service line represents the biggest one in the Hungarian market provided by independent Hungarian SME with about 150 clients.

Memolux was the prime user and contractor of the PASS (Pay-roll Accounting and Settlement System) project, which was the first Central and Eastern European ESSI PIE project directly supported by the European Commission. The PASS project was carried out under the ESSI initiative with the financial support of the Commission of the European Communities under the ESPRIT Programme EP21223.

ISCN (IRL)

ISCN was founded in 1994 and is an Irish software firm that also functions (with communication equipment) as a co-ordination office for associated consultants and firms in 11 different EU countries. ISCN has a software development subsidiary in Graz, Austria. The Austrian site co-operates with development firms such as Hyperwave and the research institutes in Graz, Austria.

ISCN in cooperation with the largest research institutes of Scandinavia runs an annual software engineering conference, and is a technical consultant and co-ordinator in different cost sharing or EU projects in the fields of process analysis, improvement, and skill evaluation. <http://www.iscn.ie>

ISCN was the technical co-ordinator of PICO (Process Improvement Combined Approach) which developed a set of training courses, a book (published by IEEE), a tool, with experience contribution from major European industry such as Siemens, Alcatel, etc. and users from SMEs (about 300) throughout Europe.

ISCN is the technical co-ordinator of the VICTORY initiative which established a

virtual organisation and e-commerce system for Software Process Improvement in Europe.

Concerning virtual team-work office applications ISCN partners have developed so far two NQA (Network based Quality Assurance) tools, one on basis of Hyperwave for cross-country cooperations and one on basis of Microsoft for Intranet team support.

ISCN is a consulting provider for major car manufacturers and suppliers and telecom solution providers in the fields of Software Process Improvement.

References

- [1] A Review of Economic Trends in Central and Eastern Europe, Autumn 2000, in cooperation of KOPINT-DATORG Co. Ltd. and Deloitte & Touche Central Europe
<http://www.deloitte.hu/pdf/53.pdf>
- [2] The E-Business Tidal Wave by Trevor R. Stewart (Deloitte)
http://www.deloitte.com/tidalwave/dtt_ebus.pdf
- [3] Business success factors: <http://www.deloitte.com/tidalwave/succeed.htm>
- [3] EurActiv CrossLingual project (Update: 04/05/200),
<http://www.euractiv.com/cgi-bin/eurb/cgint.exe/33757-617?714&1015=3&1014=inf>
- [4] The Bestregit Project , http://www.iscn.at/select_newspaper/people/bestregit99.htm
- [5] Pioneering process improvement experiment in Hungary, Miklós Biró , János Ivanyos , Richard Messnarz ,
<http://www3.interscience.wiley.com/cgi-bin/abstract/76503384/START>
- [6] http://www.iscn.at/select_newspaper/qa-systems/nqa.html
- [7] The European Observatory for SMEs : Sixth Report (July 2000, submitted by the EC)
- [8] Reporting EU enlargement - the view from both sides, An EJTA Euroreporter project sponsored by DG X of the European Commission, edited by Mogens Schmidt and Róisín Scullion, European Journalism Centre, Maastricht, April 1999.
<http://www.ejc.nl/hp/reue/contents.html>
- [9] Media innovation, professional debate and media training a European analysis, Jan Bierhoff, Mark Deuze, Claes de Vreese, European Journalism Centre, Maastricht, December 2000. <http://www.ejc.nl/hp/mi/contents.html>
- [10] The future of the printed press challenges in a digital world, Monique van Dusseldorp, Róisín Scullion and Jan Bierhoff, European Journalism Centre, Second edition, Maastricht 1999. <http://www.ejc.nl/hp/fpp/contents.html>

Experience with Team-work Concepts for Virtual Quality Management

Dr Richard Messnarz, Director ISCN Ltd., Dublin, Ireland

Rmess@iscn.ie, rmess@iscn.at

Gabor Nadasi, Rainer Bernhard, NQA Developers, ISCN, Graz, Austria

Rbernhard@iscn.at, gnadasi@iscn.at

Klaus Pfreundner, Hyperwave GesmbH, Munich, Germany

Abstract:

This paper describes experience with the implementation of a teamwork - based virtual quality assurance solution in co-operation with the company Hyperwave. The system can be adapted to different industry domains or team-work scenarios.

In an EU funded project TEAMWORK this system will be adapted for ISO 15504 compliant processes in the defense sector, for ISO 9001:2000 and ESA standards in the aerospace sector, for ISO 9000-2 in the service sector for innovation transfer agencies in different European countries, as well as for research networks to establish collaborations across different regions on focussed research topics.

The paper contains a description of the underlying principles of a virtual organisation, the paradigms followed in a system called NQA (Network based Quality Assurance) to establish such a virtual collaboration for the purpose of shared quality assurance, and presents an outlook where and how these solutions can impact the way of working.

We are grateful to the European Commission for financially supporting the TEAMWORK project.

Richard Messnarz, Dr., Florence House, 1 Florence Villas, Bray, Co. Wicklow, Ireland, ph.: +353 1 205 0020, fax.: +353 1 205 0021

Introduction into Virtual Organisations

What is a Virtual Organisation

A growing European market with an exponential growth in the communication sector leads to requirements for different types of organisations.

How quick can one react to new market demands ?

How is it possible to create a critical mass of competence from different European regions ?

How can collaborations work if people and organisations are distributed all over Europe ?

These questions do not only affect the structure of organisations, they have an impact on the team-work concepts, infrastructure requirements, and learning cultures. So further questions arise:

How can skills be assessed and trained if all teams are largely distributed ?

Which social conflicts can arise and how could they be managed ?

A **virtual organisation** is a strategy to

1. Build an infrastructure which allows different organisations to collaborate on certain projects through a (distributed) server concept.
2. Build a set of processes and quality criteria which are supported by the infrastructure and help to control the performance and quality of the distributed team.
3. Build on team-work and communication concepts which support distributed work.
4. Build information retrieval mechanisms which allow a quality manager of the virtual organisations to extract the status of the project electronically at any given time.

These aspects are discussed in the chapter about the NQA (Network Quality Assurance) concepts.

Further **training factors** to be taken into account **in virtual organisations** are to

1. Build a network based skill assessment so that people can assess themselves and produce profiles of skill gaps. E.g. to see which skills they still miss to fulfil the job of a project manager.
2. Build a set of e-Learning environments which allow the people to upgrade their skills.
3. Build a discussion environment which allows people to discuss their topics.

E.g. like an electronic meeting room.

These aspects are discussed in the chapter about the CREDIT (Accreditation over the Internet) system.

In general a virtual organisation (if properly built) leads to knowledge. An information system is a collection of data, while knowledge (like in a brain) links different elements of the information system so that for a certain context knowledge is created.

If a virtual organisations fails to build knowledge it will be a project repository rather than a virtual organisation which brings a critical mass of competence together to solve a specific problem (e.g. a project).

Typical mistakes are:

- Organisations buy a technology infrastructure and think that this is already it !
- Organisations establish processes on the infrastructure but leave out the quality issue !
- Organisations establish processes, quality and infrastructure correctly but do not use concepts that favour distributed team-work !
- Organisations do all correct but forget the social factors and the acceptance of staff !

Further **commercial factors** to be taken into account **in virtual organisations** are to

1. Build an Amazon like shopping site through customers can access the project results.
2. Build a search mechanism helping customers to find products they need easily, including guided tours on the Internet and rating and search engine facilities.
3. Build on a commercial image ("branding") which allows many partners to be commercially represented under one umbrella (even if organised electronically).

These aspects are discussed in the chapter about the Victory (Virtual Enterprise for Software Process Improvement) concepts.

The Underlying Success Principles

Process Oriented

Each process has an end-to-end responsibility to fulfil the contract to decrease the time-to-market and increase the performance all tasks for a particular project have to be organised as a process. The flow of information, work and products

are inherently combined together.

Customer Oriented

Customer satisfaction is the primary goal for each process as well as for the virtual organisation as a whole. Therefore it is necessary that each software process could conceivably be a customer for other processes as well as a supplier for their own customers.

Transaction Oriented

Between software processes, there are order-delivery relationships with clearly defined interfaces for communication and data exchange. Through streamlining the amount of interfaces should be reduced since they are sources of information loss.

Object Oriented (principle of autonomy)

Each of these processes can be seen as an object that uses resources and fulfils tasks. Objects can be described via different performance and quality parameters. These parameters are combined with metrics that are used for the selection of the best available processes for a particular virtual organisation.

Continuous Improvement

In a virtual organisation, continuous improvement occurs at the network level because the network is constantly improving the co-ordination and communication between the individual companies (nodes). And at the individual node level, each company is constantly improving its core competencies.

Virtual

Companies have to go with their products and services to the geographical location of their customer, and they have to talk the same language as them. Distributed processes are necessary to fulfil such a requirement. A virtual organisation is the result of such an intention.

By combining the core competencies of many individual companies within the network, each virtual enterprise is more powerful and flexible than its individual parts. Each company in a virtual organisation is chosen because of its process excellence. The result is a more powerful organisation since it is made up of the best available core competencies. By having all partners agree with and commit to defined schedules and costs before the start of the project, the risk is reduced to a minimum.

What is the Economic Impact

To stay competitive in today's global market, it is necessary to set up win-win

based agreements in cost sharing projects where partners from different countries share the risk and the effort, and jointly exploit ideas, products, and services. Through effective and distributed collaborations, organisations can cut down their risk significantly (e.g. sharing the development cost with other partners), and can reach a much larger market.

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

One of the principal benefits is that it provides strategies for improving service velocity - the rate at which software can be brought to market and/or customised. Through such an organisation, access to resources, know-how, and the markets of partners are available, in a way that costs and time can be saved in favour of a joint production. However, the major goal is, via an optimised information management, to increase flexibility, productivity and customer orientation [18]. Quality itself is not the primary goal of a virtual organisation, rather it is accepted as a necessary condition for maintaining competitiveness [23].

Collaboration through a virtual office also implies a need for a knowledge base that can be shared between organisations. The difference between information (as it is offered now by many Web servers) and knowledge is that knowledge is created from information by putting a structure onto the information so that it can be shared, multiplied, and understood across a team. Not only the information and its structure is relevant, but also meta-information like owner, creation time, or when the information was last retrieved and by whom. This information about information is also necessary for turning information into knowledge.

This may also have a strategic impact for the European Union in general. Under the 4th framework program the ESSI (European Systems and Software Initiative) initiative funded hundreds of PIEs (Process Improvement Experiments) at smaller and medium sized companies across Europe to improve their development capabilities and software processes.

The 5th framework program supports the virtual information society strategy so that technologies are sought that could connect those efficient companies into focused collaborations building the strengths together as if they are a big company.

In a Europe where most industry is small and medium sized, such a strategy could create competitive advantages against other countries and continents.

What are the Functions

Whereas the virtual organisation's life span is limited by the software project's life cycle, there is need for a permanent and preferable flat organisation that provides for the availability of required competence resources, optimised communication channels and definition of a meta process facilitating co-operation and interaction. Therefore, management activities encompassing all software processes are needed to configure the virtual organisation, to co-ordinate their co-operation, and to conform them to a common strategy.

Distributed collaboration requires effective co-ordination between the involved partners' work and quality control mechanisms. This can be addressed with by a virtual office on a network that includes project archives and document management, configuration management, guide-lines and computer support for project documentation, network and computer supported information flow, and appropriate security mechanisms assuring privacy of the materials exchanged and produced. In addition such a system needs a flexible access control and authentication system to manage the access to all information for every individual on the network.

However, virtual organisations are not limited to just such quality assurance functions, they can also offer a vast array of additional services like customer support, project management, component and other administrative functions, where quality assurance is a core component.

The majority of communication in a distributed collaboration uses asynchronous mechanisms (e.g. email, web-publishing and retrieval) but there is also need for synchronous communication like chat or telephone conversations. The information which is included in synchronous communication should also be archived by the information infrastructure. The integration of information- and telecommunication systems seems to be necessary.

Requirements for successfully applying the concept of a virtual organisation to the software production process are among other things an open and standardised information infrastructure, defined software processes, confidence in the performance of all partners involved, the participation of all partners in the decisions and the overall result, as well as a modular software architecture.

Individual companies can be geographically distributed and therefore use different languages, and different legal and social systems. The connecting information and communication technology is charged with selecting, measuring and controlling the processes in spite of these constraints. The underlying technologies have to meet the requirements for knowledge and information storage, for de-centralised information access and retrieval, as well as for the short-term merging of distributed knowledge [19].

Information Infrastructure

However, most of the requirements that virtual organisations demand (in terms of information infrastructure, especially technical openness, distributed storage of data, and security mechanisms) can be fulfilled with existing information and communication technologies [18]. Today, the most powerful and cost-effective infrastructure for enabling virtual organisations is the Internet [3]: with its failsafe network topology as the communication infrastructure and the different Internet services like email, WWW and ftp, as the information infrastructure.

Via virtual private networks virtual organisations can use the Internet as an Intranet. The information (e.g. project documentation, customer data) that is necessary for performing the business is presented by application servers to all the members of a virtual organisation. Such a Web-based application server is the Hyperwave Information Server - a broad and feature-rich application development platform that has been used to build web-based applications in areas that are important for virtual organisations: knowledge management, document management, Web-based training, project management, and many more. This flexibility and customisability is one of the major strengths of the Hyperwave Information Server.

To meet the requirements discussed above for a virtual organisation, flexible IT-systems are necessary. They have to be quickly adaptable (like the plug&play concept) to new processes and IT-system. As an example for such a system the Hyperwave Information Server supports a virtual organisation and its processes with a great deal of built-in functionality including the following:

- A dynamic structure for the presentation of documents and links that allows customised views of information
- A clear separation of information and its presentation
- Documents are stored as objects containing information, metadata and functions
- A built-in user and group based security mechanism (access control)
- A scalable architecture for connecting many Hyperwave Information Server together into one server pool
- A channel mechanism that supports passive information retrieval (notification)
- Users can create new information through linking documents together
- Collaborative authoring is supported with integrated document versioning, locking, and configuration management
- Object-Oriented programming methods allow the development of new applications and/or the extension of existing functions

Combining the concept of an virtual organisation with such an infrastructure leads to following simplified scenario:

Each process (or company) runs its own information server for their special tasks. Process dependent applications control the input and output of data. All servers in an virtual organisation are combined into a server pool where each member can access information from the other. But some information should not be accessible to the whole virtual organisation. Therefore the access rights must be restricted to the information that is defined in the general agreements. These contracts also specify the overall workflow and the interfaces between processes. Information that should be exchanged is linked from the server that holds the information to the server where the information is needed without copying it. This reduces the amount of storage, increases the maintainability of information (only one source), and simplifies the access control. After a common project is finished, the servers are removed from the server pool and all links between servers are automatically removed or disabled.

The features and the scenario described above are only a small overview of what is possible when a Web-based server act as an information infrastructure for a virtual organisation. Better is a real world example of an actual implementation for a specific project. NQA is such an example. It supports the quality management in a virtual office in conjunction with ISO 9001.

NQA (Network based Quality Assurance)

Paradigms Underlying the NQA Concept

The NQA approach bases on three principles which have been discussed and published at previous ISCN conferences (<http://www.iscn.ie/conferences>) and about which there is a book being published by IEEE [13]: Better Software Practice for Business Benefit - Principles and Experience (ed. Richard Messnarz, ISCN).

The three principles

Role and information flow based team work process management

Development by configuration

Re-Use pool concept

are discussed below.

A major feature to make such a virtual approach applicable for different environments is that such a system must be kept completely configurable. The menu, the data, the functions, the document/information flows can be configured for different user scenarios and this high configurability is the major feature of an NQA virtual office. It is based on standard Internet languages and scripts and on the Hyperwave information server.

The Underlying Management Principle

A software process is not seen as just a sequence of tasks with a planned result [10], but it is the result of an integrated team work environment [14]. The organisation is broken down into work scenarios (management use cases, e.g. scenario for planning, scenario for design, scenario for marketing, etc.) and each scenario is designed with

Roles who have responsibilities

Work steps to which roles and resources are assigned

A network of work steps forming a work-flow

Results produced by roles performing a certain work step in the work flow

The new approach is to think role-centered, so that by staffing of roles work scenarios in an organisation are initiated.

The advantage of the new approach is

People know their responsibilities better and know their communication interfaces to other members in the team

New staff can easily be integrated (assign a role, learn the skills required to play the role, follow the communication flows in the team)

Information technologies like NQA (because the communication interfaces become visible) can be used to support the team communication, documentation, and configuration of results.

Benefits Measured

Experiments with this approach have been carried out since 1993 at firms in Austria, Germany, Spain, and Ireland, and 7 other countries. Results are [13], [14]

A 50% reduction in effort in new staff integration

A 67% higher team motivation for using documentation efforts like ISO 9001 (share the work in a team in a defined way)

A 67% reduced maintenance and 50% higher productivity because a decomposed role based team with clear responsibilities allows good distribution of tasks (parallel and not sequential work) and avoids monolithic program architectures (all are responsible for the same software without clear distinction of interfaces and modules).

Management Steps

- Define the roles
- Identify communication flow between the roles
- Formalise communication flows (only where necessary) and define results (exchanged between roles)

The work-flow, after that, is just a waste product of the team-work model

Example from a planning scenario at Hyperwave

For each scenario there is an underlying role play clearly describing the roles played in a team, the responsibilities, and the communication flows. These communication flows result in a number of work instructions describing the roles' duties and the sequence of work steps to be performed. The same working instructions are then used, for instance, to show compliance with working instructions required by ISO 9001 [11], [12].

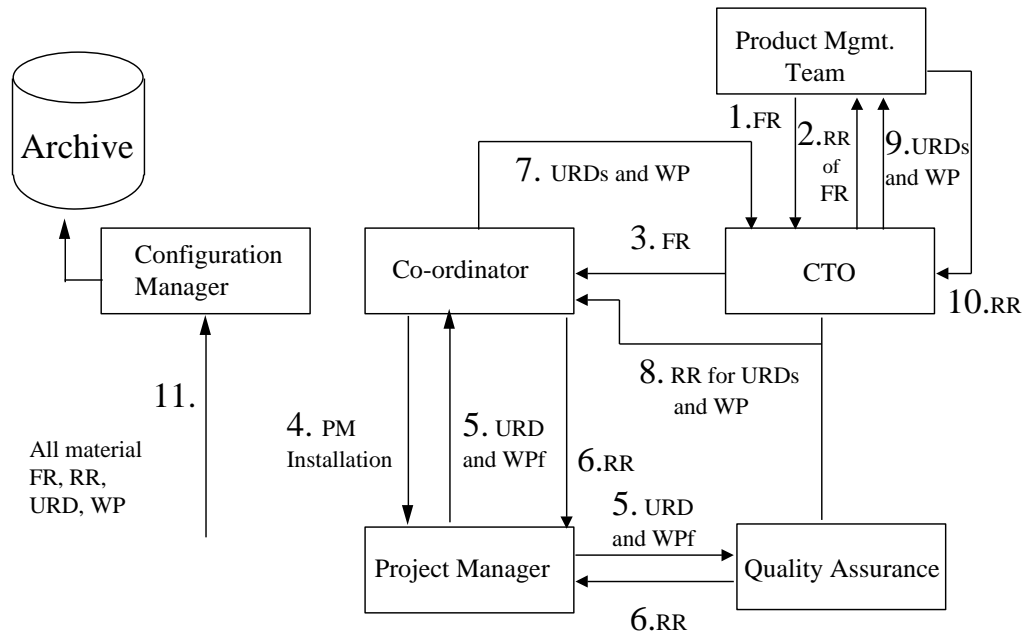


Figure 1: A Role Play for Feature Request Management and Planning

Work Instructions for the Feature Request and Planning Scenario at

Hyperwave

1. The Product Management Team (PMT, customer) makes a Feature Request (FR). The Chief Technical Officer (CTO) receives it and archives it.
2. The CTO reviews the features together with the PMT resulting in Review Reports (RR) for the feature request and decisions about their implementation.
3. The refined feature request (for which an implementation was decided) is forwarded to the Co-ordinator (CRD).
4. The Co-ordinator assigns the feature request to a responsible project manager. For each release there are many such feature requests so that the previous steps are repeated many times.
5. The responsible Project Manager (PM) draws up a draft User Requirements Document (URD) and a URD specific Work Plan (WPprj), and forwards the draft for review to the Quality Assurance (QA) and the Co-ordinator (CRD).
6. The draft URD and WPprj are at the same time reviewed by the Quality Assurance (QA), and the Co-ordinator (CRD), resulting in Review Reports (RR).
7. The Co-ordinator approves the WPprj and combines them into an overall Work Plan (WP) for the organisation, and forwards all URDs and the overall WP for review to the CTO.
8. CTO approves the URDs and the WP.
9. PMT receives URDs and WP for final review.
10. PMT reviews and gives acceptance to the URDs and the overall WP.
11. Configuration Manager (CM) controls that all materials produced in the work flows have been properly archived. Special care is taken on the trace-ability between feature requests, requirements in the URD, and proposal/agreement issues.

Only after the establishment of such a role-based model the information flows become clear and a tool can start to support the team communication and quality control activities through a virtual office of distributed competence teams.

The Information Technology Principles Underlying an NQA Concept

Development by Configuration

This paradigm bases on the fact that functionality is to be separated from data, and that data can be assigned with functionality by the user through configuration. NQA concepts must developed according to this principle and

allow each organisation to insert their own documentation or result templates, and the NQA system then automatically generates (with the creation of objects from the templates) the functionality to the created objects.

This way users can insert and maintain document or result templates and adapt the system to their own specific documentation requirements without any change or customization of code (just by configuration of data).

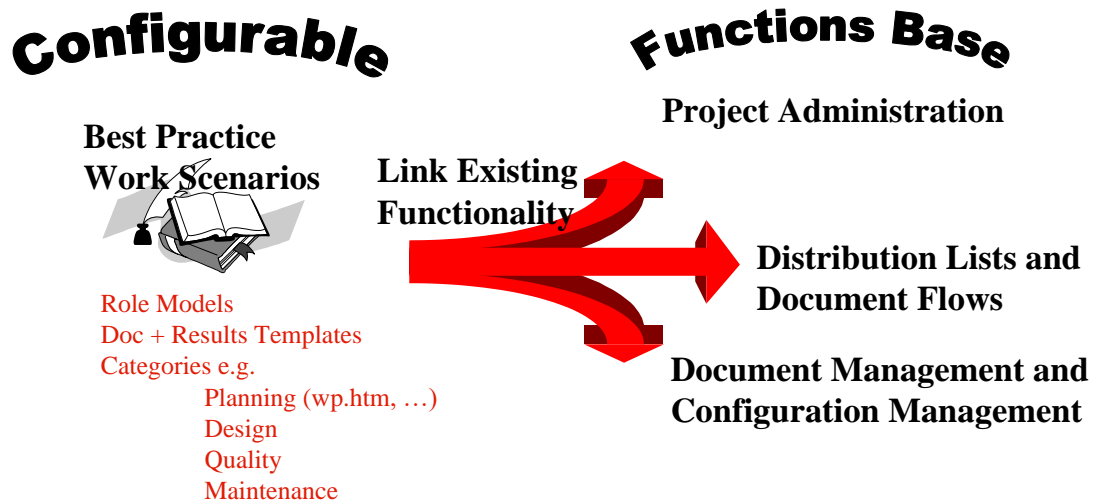


Figure 2: Data and Functional Configurability

Function Base Driven Configuration (Re-Use Pool Concept)

At the moment three basic elements can be configured to which the above functionality is generated.

Documents - The below picture shows the standard window for document creation, with SAVE the functionality is generated to the template taken from the pool and a first version is issued under configuration management)



Figure 3 : Document Object Creation Window

Reports - The below picture shows the standard window for report creation,

after ADD a report is added to a list and the functionality is generated to the template taken from the pool and a first version is issued under configuration management.

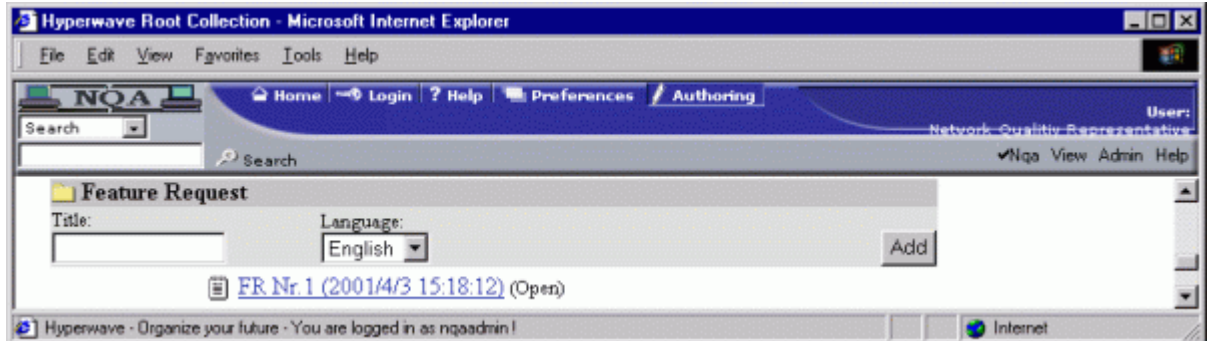


Figure 4 : Report Object Creation Window

Linked Reports - same as with reports, plus the report is automatically linked backward and forward to what has been selected in the right combo boxes.

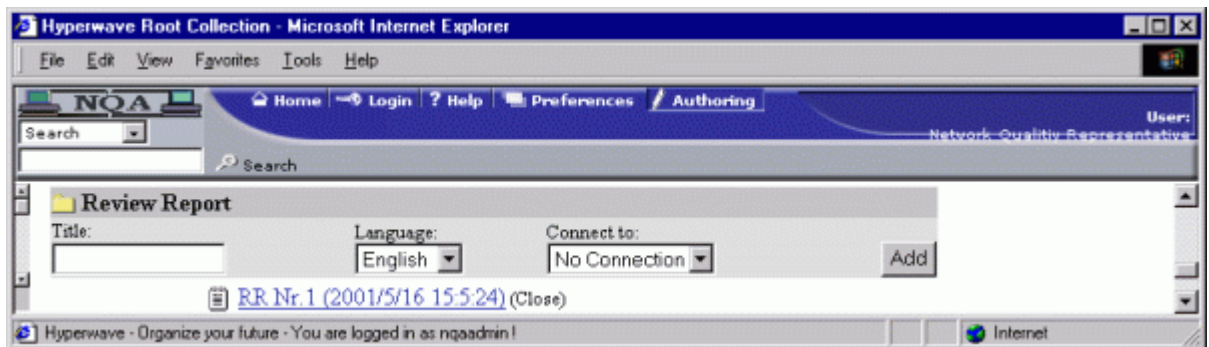


Figure 5 : Linked Report Object Creation Window

Depending on the user needs the three elements are configured. E.g. Linking Feature Requests (FR) with user Requirements Documents (URD), so that an URD is automatically created by the links to accepted FRs (example from a customer wish from Daimler Benz).

Further basic elements might be considered and inserted into the NQA configuration pool in later releases.

How an NQA Virtual Office Works

A required functionality of an NQA system comprises the automatic assignment of the following functionality to created objects -

Document Management

Creation of documents from a template pool (configurable by customer). Automatic administration within a project structure under a certain documentation category (e.g. planning document). Electronic submission to a distribution list (workflow). Version management and change control (see configuration management). Automatic forward linking to reports (e.g. a number of Review Reports linked forward and back to the document version, see link management). Download, edit, and publication facilities. Computer supported test status.

Report Management

Creation of reports from a template pool (configurable by customer). Automatic administration within a project structure under a certain documentation category (e.g. quality control reports). Electronic submission to a distribution list (workflow). Version management and change control (see configuration management). Automatic forward and backward linking between documents and reports, or reports and reports (e.g. linking test protocols with problem reports, and problem reports with modification reports). On-line edit of forms at server side, and on-line submission (no download necessary for edit).

Workflow Management

Electronic submission of reports and results to team members. Encryption module can be used. Administration of distribution lists (for automatic forward). A communication log per project archiving all communication flows between team members (roles).

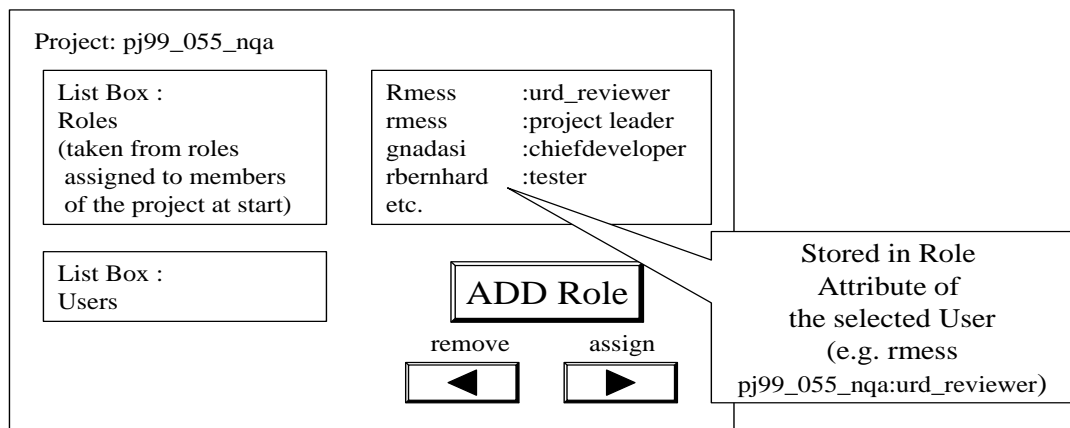


Figure 6 : Configuration of Roles and Users

People are assigned to project groups as well as to roles defined in a particular project, and distribution lists are automatically generated so that documents and reports can be automatically submitted to the appropriate team members who are responsible for specific role(s) in the project group. Once a document has been submitted it will be

checked in into the system with a new version number and in the *task collection* of the appropriate member a new task entry will be inserted.

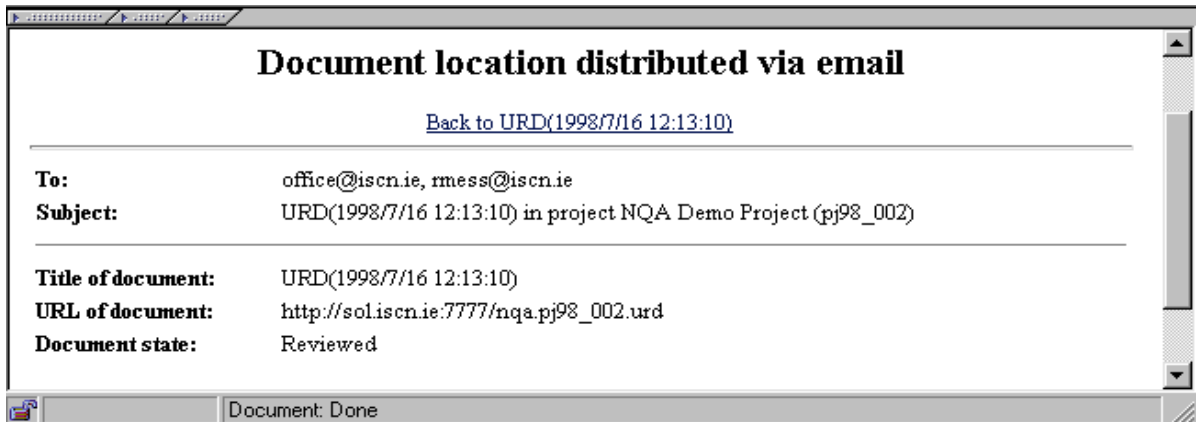


Figure 7: A Standard Notification Message for Submissions

The project members task collection in turn contains one separate *project task collection* for every project that a particular person is participating in. Finally a project task collection contains the list of tasks that a team member has to carry out within the framework of a particular project. The “NQA Users Collection” the “NQA Users Task Collection” and the collection called “Tasks for the project: XYZ” will be automatically created and maintained by the system on demand.



Figure 8: The collection “NQA Users Collection” contains one coll. for each user of NQA



Figure 9: NQA Users Task Collection

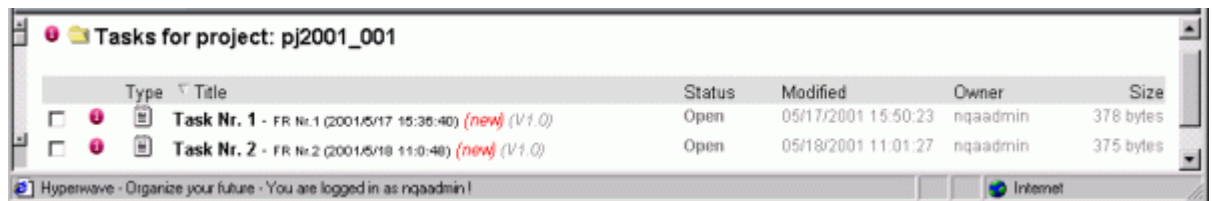


Figure 10: Task list of the user “nqauser01” participating in the project “pj2001_001”

Configuration Management

Version management. Registration of versions in a document (result) history. Check-in and Check-out functions. Revert to previously archived versions. Test status information in document history.

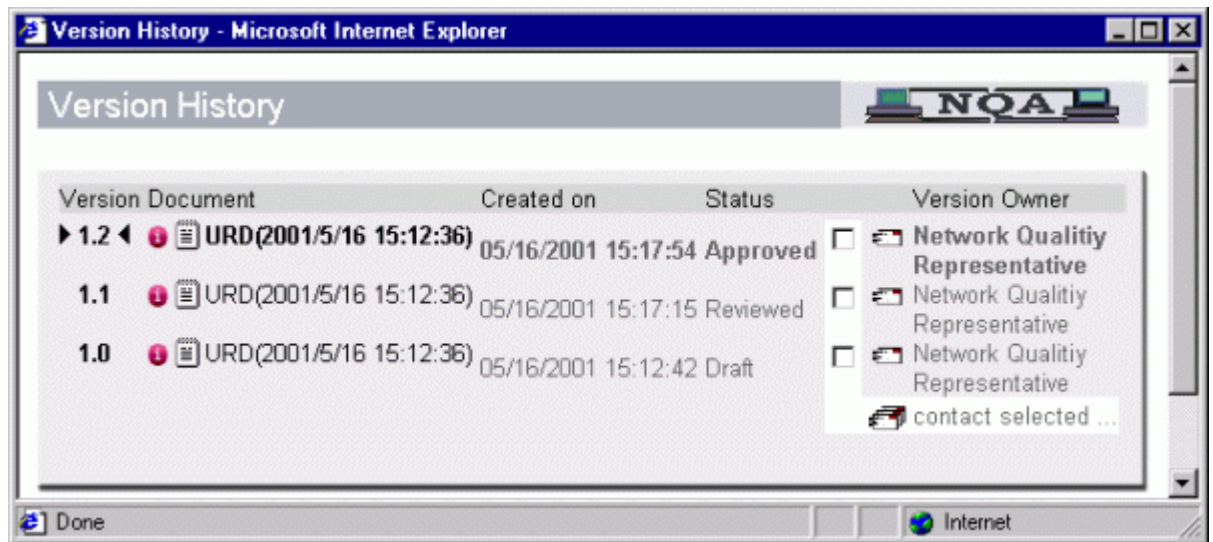


Figure 11: Version control with Object History Including Test Status

Link Management (Forward and Backward Tracing)

Definition (see functional configurability) of links between report and document types. Automatic assignment of linking properties to created objects. Automatic forward and backward linking according to the defined functional configuration (configurable by system administrator). E.g. linking review Reports with documents, so that by a click you switch between the document and the related reports.

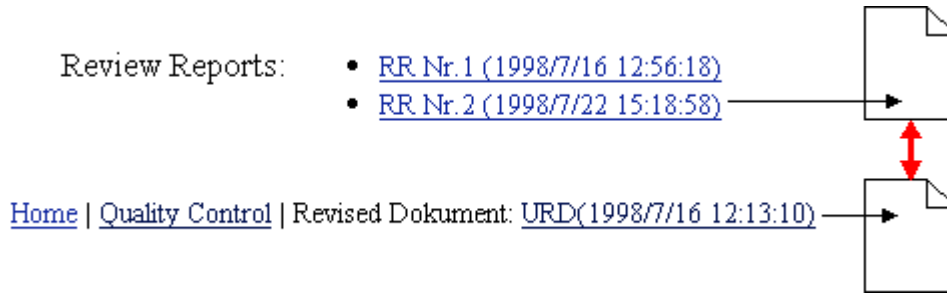


Figure 12: Forward and Backward Linking (e.g. Review Reports Linked to a User Requirements Document)

User Administration

Administration of a team per project (see only their project). An NQA system administrator nqaadmin (sees all). Administration of a distribution list (for electronic submission) per team (and in future role based per document).

On project start roles must be assigned to project members. After doing this documents can be assigned using the dialog depicted on Figure 13 to corresponding roles so that on submission a particular team member will be automatically notified on the delivery of a document that belongs to his/her field of work.

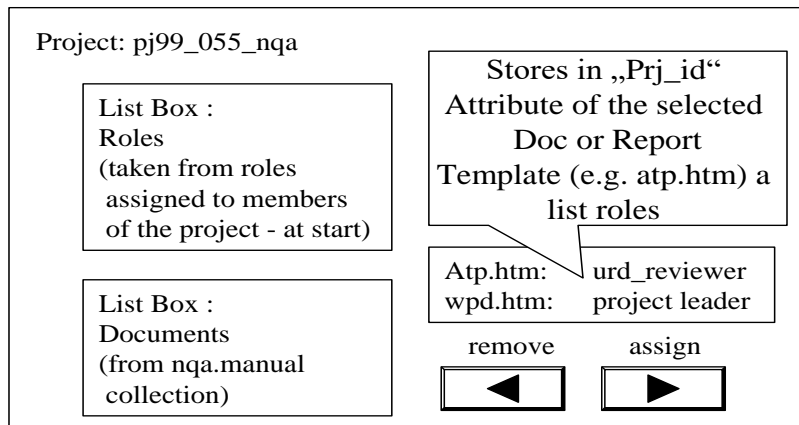


Figure 13: Configuration of Roles and Documents/Reports

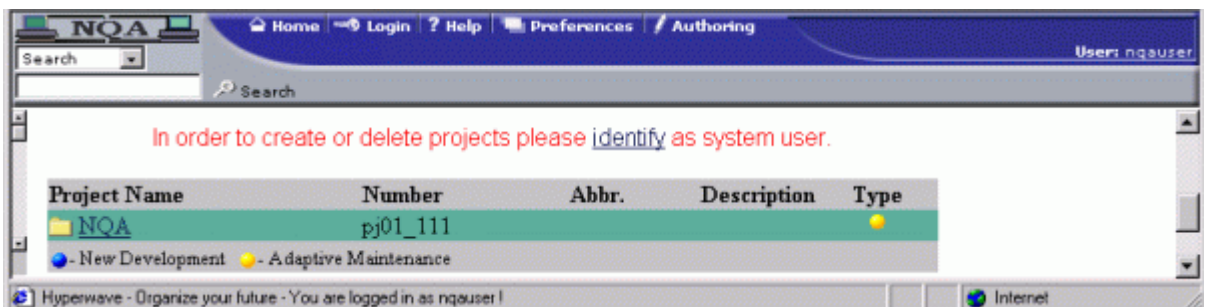


Figure 14: Identification and User Control

Project Progress or Document Status Evaluation

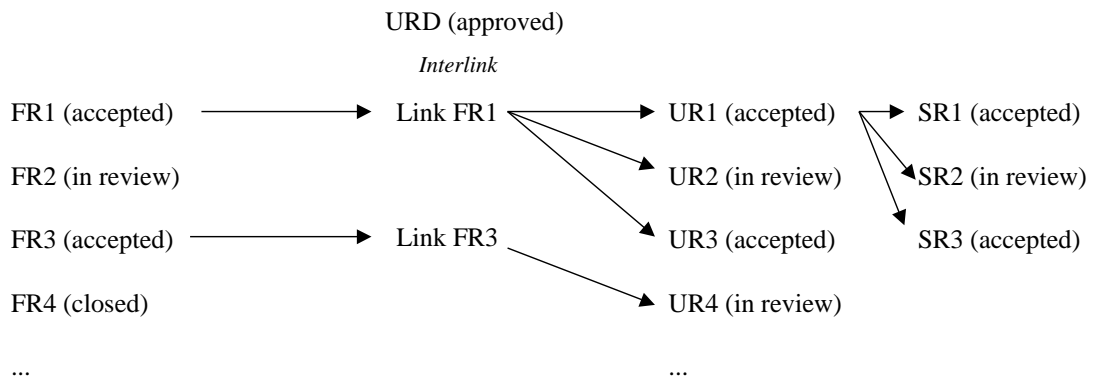


Figure 15: Project Progress Evaluation or Document Status Evaluation

By keeping track of the document status and following the links that connect documents to each other we can derive and represent data and answer questions on *project progress* e.g. ‘How many of the FR (Feature Request, status accepted) are in status SR (Software Requirement, status accepted)?’ or on *document status* e.g.: ‘How many FRs were received and how many of the FRs are closed?’ or ‘How many of the SRs are in status “reviewed“?’.

Security Management

No access without identification possible. The information in the electronic submissions contains only links to info at the server which requires identification. If even these links should be protected an additional encryption module from Hyperwave can be installed.

By just using Netscape the team members (from home, from any work place, etc.) can access the NQA virtual server and work on-line through a joint interface.

The ISO 9001 Experience Pool

NQA is delivered together with a complete electronic ISO 9001 manual, with examples and role plays and with templates in German and English for all required ISO 9001 documentation.

It is also delivered with implemented procedures and scenarios to run ISO 9001 compliant planning, design, delivery, and maintenance.

This comprehensive set of information allows organisations to achieve an ISO 9001 certificate much easier.

It also gives (through its configurability of a template pool) organisations who have already an ISO 9001 certificate a huge support in shifting from a paper based ISO 9001 environments to a fully electronic based ISO 9001 computer supported system,.

NQA systems at organisations in Austria, Germany, and Hungary have already been certified by TÜV, Norske Veritas, and ÖQS.

NQA's Future Plans

NQA is currently being adapted to three industrial domains, including defense, aerospace, innovation centres, and research networks. Results in these domains will be available in summer 2002. For the moment it is used for ISO 9001 compliant systems and organisations with NQA systems have been certified by different certification bodies: DNV, ÖQS, TÜV.

NQA is an application developed by ISCN on top of the Hyperwave information server. It is planned that NQA in future is encapsulated as a virtual quality assurance solution for ISO 9001 under Hyperwave, and sold on a CD to all Hyperwave users.

Future Outlook into Virtual Organisations

As an organisation form the co-operation network virtual organisation is very suitable for the development of software [3]: not only to decrease costs, to be able to react to rapidly changing situations in a flexible way and to distribute risks, but moreover to increase customer benefits by ensuring performance and quality. To achieve this, it is necessary to understand the underlying methods and processes, especially because of the already existing quality problems in software development [4][9].

The quality of the whole company is a condition for the quality of the products they produce - the quality of the processes and the quality orientation of the company culture, as well as the quality of the employees and the management [16]. Because of the temporary nature of the co-operation no identification with the virtual organisation will evolve, and therefore no company culture as well. The loyalty to the company will be replaced by the loyalty to the product.

It is also a disadvantage that the performance of the Internet is not as good as it

should be, because of insecure parts of networks and slow transmission rates. A final area of problems is represented by the existing legal vacuum, especially in terms of the validity of legal documents in electronic form, the acceptance of electronic signatures, patent rights and international product liability [18].

References

- [1] Arnold, O., Faisst, W., Härtling, M., Sieber, P., Virtuelle Unternehmen als Unternehmenstyp der Zukunft?. HMD – Handbuch der Datenverarbeitung, 32 185/1995, pp.8-25.
- [2] Cusumano, M., Japan's Software Factories: A challenge to U.S. management. Oxford University Press, New York, Oxford, 1991.
- [3] Gao, J.Z., Chen, C., Toyoshima, Y., Leung, D.K., Engineering on the Internet for Global Software Production, IEEE Computer, May 1999, p.38-47.
- [4] Gillies, A.C., Software Quality: Theory and management. Int. Thomson Computer Press, London, 1992.
- [5] Hammer, M., Champy, J., Reengineering the Corporation. Harper Collins Publishers, New York, 1993.
- [6] Handy, Ch., Trust and the Virtual Organization. Harvard Business Review, May-June 1995, pp.40-50.
- [7] Herzwurm, G., Mellis, W., Schmolling, K., Software Factory - Ein Statusbericht. HMD – Handbuch der Datenverarbeitung, 180/1995 p.8ff.
- [8] HPO, High Performance Organizations Handbook. Technical University Graz, 1996.
- [9] Humphrey, W.S., Managing the Software Process. Addison-Wesley, Reading, 1989.
- [10] IEEE *Software Engineering Standards Collection*, IEEE Standards for Software Quality Assurance Plans (IEEE 730-1989), Quality Assurance Planning (IEEE 983-1986), Project Management Plans (IEEE 1058.1-1987), Configuration management Plans (IEEE 828-1990), Software Verification and Validation Plans (IEEE 1012-1986), IEEE Computer Society Press, 1991
- [11] ISO 9000-3. Quality management and quality assurance

standards. International Standard. Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply and Maintenance of Software. ISO (1990).

[12] ISO 9001. Quality Systems. Model for Quality Assurance in Design/ Development, Production, Installation and Servicing. International Organisation for Standardisation, Geneva (1987)

[13] Messnarz R., Tully C. (eds.), *The PICO - Book: Better Software Practice for Business Benefit - Principles and Experience*, IEEE Computer Society Press, in publication

[14] Messnarz R., Stubenrauch R., Melcher M., Bernhard R., Network Based Quality Assurance, in: Proceedings of the 6th European Conference on Quality Assurance, 10-12 April 1999, Vienna , Austria

[15] Marciniak, J.J., Reifer, D.J., *Software Acquisition Management*. John Wiley & Sons, New York, 1990.

[16] Mellis, W. Herzwurm G., *Qualitätsmanagement*. Business Computing, 8/1994, p.36ff.

[17] Mellis, W. Herzwurm G., Stelzer, D., *TQM der Softwareentwicklung*. Vieweg, Wiesbaden, 1996.

[18] Merkle, M., *Virtuelle Organisationen – Ihr Erfolgspotential: eine integrative Informationsinfrastruktur*. Institutsbericht des IFI der Universität Zürich, 1996.

[19] Mertens, P., Faisst, W., *Virtuelle Unternehmen, eine Organisationsstruktur für die Zukunft?*“. *Technologie & Management*, 44 2/1995, p.61ff.

[20] Mertens, P., Faisst, W., *Virtuelle Unternehmen nutzen weltweite Netze*, 1996.

[21] Picot, A., Reichwald, R., Wigand, R.T., *Die grenzenlose Unternehmung – Information, Organisation und Management*, 2. Auflage. Gabler, Wiesbaden, 1996.

[22] Scholz, Ch., *Die virtuelle Organisation als Strukturkonzept der Zukunft?* Arbeitsbericht Nr.30, Lehrstuhl für Betriebswirtschaftslehre, Saarbrücken, Universität des Saarlandes, 1994.

[23] Tapscott, D., *The Digital Economy*. McGraw-Hill, New York, 1996.

[24] Zesar, K.D., Zechner, M., Salhofer, P., Schuster, G., Muelleitner, G., Performance and Quality Aspects of Virtual Software Enterprises. In: Proc. of the 24th EUROMICRO Conference, Västeras, Sweden, August 1998.

[25] Zesar K.D., Mesaric G., A Process-Oriented Quality Management Model for Software Developing Cooperation Networks. Proc. of the 6th European Conference on Software Quality, Vienna, April 1999.

SPI repository for small software organisations

Marion Lepasaar, Timo Varkoi and Hannu Jaakkola
Pori School of Technology and Economics, Pori, Finland

Abstract

Small organisations require assistance to become involved in software process improvement (SPI). Difficulties in using international SPI models could be overcome with a collection of appropriate tools and information that altogether form a learning and support environment. Our experiences have motivated us to collect necessary tool and methods to support SPI activities. In this article we describe the SPI repository which is the central part of the environment and which provides necessary data for various process improvement related activities.

Introduction

Using SPI can improve the way that an organisation develops, maintains and delivers software systems to achieve real business benefits. SPIN (UK) has recognised that its members need support with capability evaluations, process improvement and the application of international SPI models such as ISO 9001, CMM, SPICE and Bootstrap etc. [1]

The most common starting point for SPI in a small company is that they have realised the need for process improvement, and have understanding and experience of good practices. On the other hand they do not have an overall view of the company's situation and do not know any process models. Some companies may have started to implement a quality system and through it they have analysed the processes, but nevertheless do not know how to improve processes effectively. [9]

Our experiences in using an international software improvement model in small software organisations generated the idea to collect the tools and methods needed to successfully start up software process improvement. Small software organisations, in their feedback of a regional SPI project in Finland, found the SPI training and external consultations to

be the key success factors of SPI activities.

Since the supporting resources are limited the need to reduce the costs and lower the barriers to start up SPI is obvious. The idea of collecting the tools and methods can be extended to the development of an environment that supports effectively both the SPI work and the related learning process. The environment should include models, methods, tools and information in order to benefit software process improvement. [10]

The goal of this paper is to describe the content requirements for the repository, which forms the central part of the SPI environment. The repository should hold information or reference to the information necessary for efficient software process improvement activities.

Introduction to RaSPI

In order to support small software organisations in their effort to continuously improve their processes, we aim to create a support and learning environment for SPI, called rapid software process improvement support environment (RaSPI). RaSPI aims to provide the necessary external assistance in a portable environment to support the SPI work in small software organisations. [10]

The environment can be divided into three parts as seen in Fig. LEPASAAR.1:

1. The continuous improvement process of SPI
2. The application views of the RaSPI repository
3. The RaSPI repository

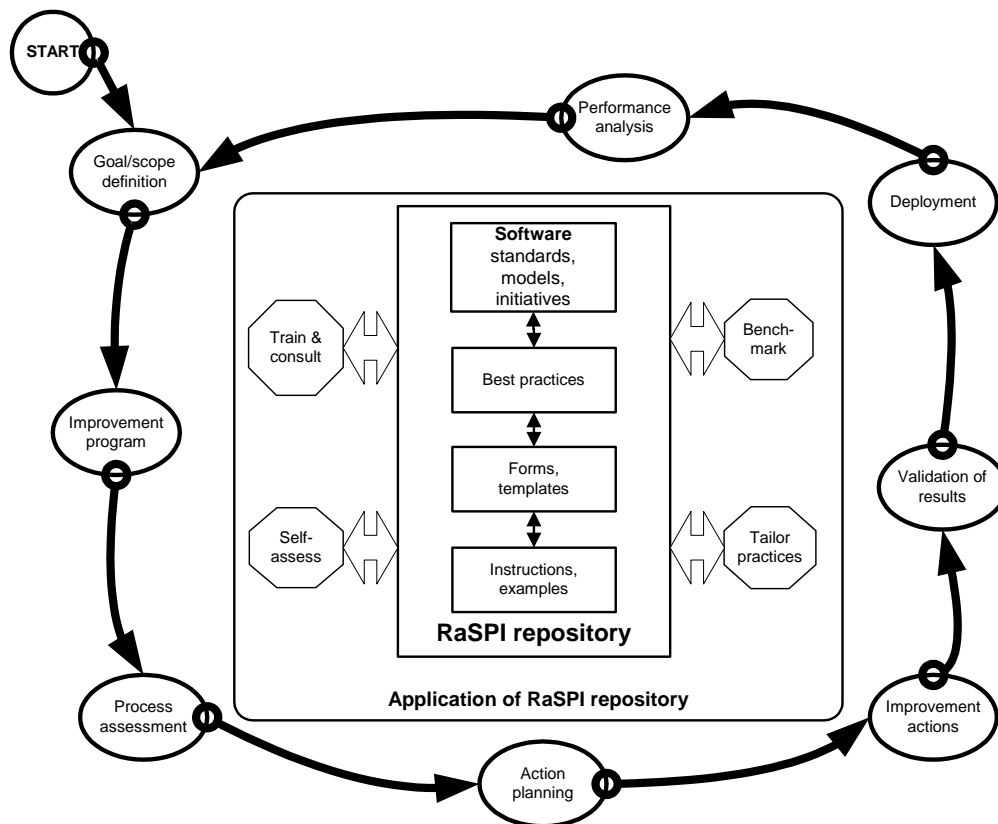


Fig. LEPASAAR.1 : The RaSPI environment

RaSPI repository plays the central role in the environment providing the data of different software models and standards for SPI with best practices, forms, templates, instructions and examples.

There are several possibilities for the application of the repository. It can be used for self-assessment purposes, training and consultation, benchmarking or for tailoring the software engineering practices of the software organisation.

The repository should support continuous software process improvement [4], which is described in the eight steps around the repository on Fig. LEPASAAR.1. Depending on the needs of the organisation the data stored in the RaSPI repository can be applied to support any of the activities within these improvement steps.

In this article we list the minimum requirements for the content of the repository. The next chapter defines the original requirements for the contents of the repository.

RaSPI repository

Repository, as viewed in the context of the RaSPI environment, is an organised data storage that contains the necessary data or reference to the data of models and methods needed in either software process improvement activities, implementation of software-engineering practices, software-engineering related training or self-assessment purposes (Fig. LEPASAAR.2).

Based on software process improvement process, various ways of applying the SPI-related information and our experience of SPI work in the small software organisations during the SataSPIN project [9], the repository should include at least the following:

- ✓ Software process improvement related international standards and models.
- ✓ Best practices – reference practices (descriptions of the defined and documented processes).
- ✓ Forms and templates for software engineering and software process improvement activities.
- ✓ Examples and instructions of the usage of the repository and the data it includes.

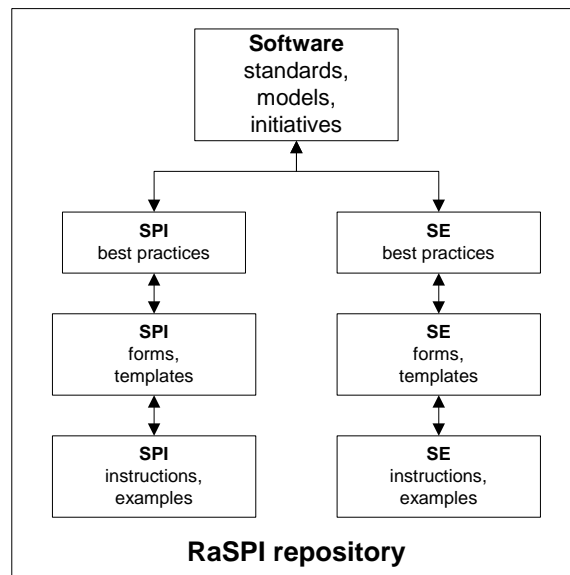


Fig. LEPASAAR.2 : The RaSPI repository

Software standards and models

A standard is a rule or basis for comparison that is used to assess the size, content, value or quality of an object or activity. There are standards that describe the nature of the object and the standards that define the way the work is to be performed [2]. Standards

are usually developed under the auspice of, and are owned and maintained by either the International Organisation for Standardisation or national standardisation organisations. The main characteristic of a standard is that it defines the minimum that has to be achieved [11].

Specialised organisations or industry consortia have developed most of the popular software process improvement models. Models are usually developed through sponsorship by a large “buyer” organisation or through an initiative by a body of companies. Models are flexible and can be customised according to the guidelines issued by the developers. [11]

For each software model or standard described in the RaSPI repository, the following should be mentioned:

- ✓ the full name of the standard, the year of release, the organisation(s) involved in creating the standard,
- ✓ short overview of the standard, including the focus area, the application purpose, and the structure of the standard,
- ✓ the reference to the full standard.

Best practices

Best practices can be divided into two: software engineering (SwE) best practices and software process improvement (SPI) best practices. Best practices, as viewed in this article, are the generally known ways to best carry out software and management activities or tasks.

SwE best practices are the reference practices that describe the activities or tasks contained in a software process, which is any process used by an organisation to plan, manage, execute, monitor, control or improve software related activities [5].

Best practices related to SPI can be viewed as the series of steps or specific improvement activities that guide through continuous cycle of improvement and are closely related to the SPI model or standard the organisation has chosen.

Forms and templates

Forms and templates ease the development of documents to record organisation’s software processes or their improvement.

The forms and templates of software engineering are related to the SwE best practices, as they form the bases to quality requirements of these practices providing the necessary checklists and document templates.

SPI forms and templates are related to the model and method the organisation has chosen for SPI activities. The SPI templates include templates for documents such as software process assessment reports and improvement plans.

Examples and instructions

Instructions for the repository could be associated to software engineering in general or to the software process improvement.

Instructions related to SwE include the guidelines for applying SwE forms and templates and implementing the SwE best practices.

Instructions related to the SPI include guidelines for choosing the SPI model most appropriate to the organisation and for applying the model chosen. There are several criteria that should be listed for choosing the appropriate model for a software organisation.

Examples should visualise the application of the RaSPI repository for different purposes described as the application views of the repository in the Fig. LEPASAAR.1.

Example of the contents

This chapter gives some examples of the contents of the repository based on ISO/IEC15504 software process model, following the list of requirements described in the previous chapter.

The software process model

ISO/IEC TR 15504 also known as SPICE (Software Process Improvement and Capability dEtermination) is an international software model. Technical reports were published in 1998-9 by International Organisation for Standardisation. SPICE is mainly used for software process improvement purposes, with the specific focus area of the process assessment.

SPICE documentation consists of 9 parts, where Part 2 and Part 3 are normative. Part 5 provides an exemplar model for performing process assessments and it is based upon and directly compatible with the reference model in Part 2. The assessment process must be documented and should be based upon a method in line with requirements defined in Part 3 and following the guidance provided in Part 4.

The technical reports are in the following nine publications of ISO/IEC TR 15504:1998 Information technology – Software process assessment:

- Part 1: Concepts and introductory guide;
- Part 2: A reference model for processes and process capability;
- Part 3: Performing an assessment;
- Part 4: Guide to performing assessments;
- Part 5: An assessment model and indicator guidance;
- Part 6: Guide to competency of assessors;
- Part 7: Guide for use in process improvement;
- Part 8: Guide for use in determining supplier process capability;
- Part 9: Vocabulary.

Best practices

SPICE Part 5 contains the list of good software engineering practices [3]. In the SPIRE project, the base-practices described in SPICE Part 5 have been viewed as the industry best practices. The industry best practices represent the general lessons learned by the industry about how to best carry out business [6].

SPI best practices can be viewed as process improvement activities or series of steps that form the improvement cycle proposed in SPICE Part 7. SPICE Part 7 provides a framework for implementing improvements in a continuous manner [4].

Forms and templates

The software engineering related forms and templates can include any templates and forms created and used in software projects by an organisation. These forms could be

checklists for software engineering practices or document templates for all the necessary software project documentation.

The SPI related forms could be the assessment forms that help to carry out process assessments. The templates could be e.g. the assessment plan templates and assessment report templates [8].

Instructions and examples

The SPICE model itself does not provide instructions for the implementation of the software engineering or software improvement activities. The instructions are created to help small software organisations in their effort to establish continuous SPI. We suggest the assignment of the SPI roles and responsibilities to clarify who in the organisation should be responsible for which specific part of the SPI work.

Along with the guidelines there should be examples of companies applying the best practices and models for software process improvement. The scheme viewed below presents an example of how a SPI program can be implemented with limited resources. The scheme is an example how improvements, based on assessment findings, can be achieved in a small software organisation by applying SPI with actual software projects. In this case the time needed for implementing improvements is relatively long. The example is derived from an actual experience of a small software organisation implementing process improvement.

The scheme followed by the small software company is presented on Fig. LEPASAAR.3. The improvement period can be divided into three phases:

1. The software projects A and B and the first assessment;
2. Analysis of the first assessment results and derivation of improvement plan during the software project C;
3. Implementation of the improvement plan and second assessment during software project D.

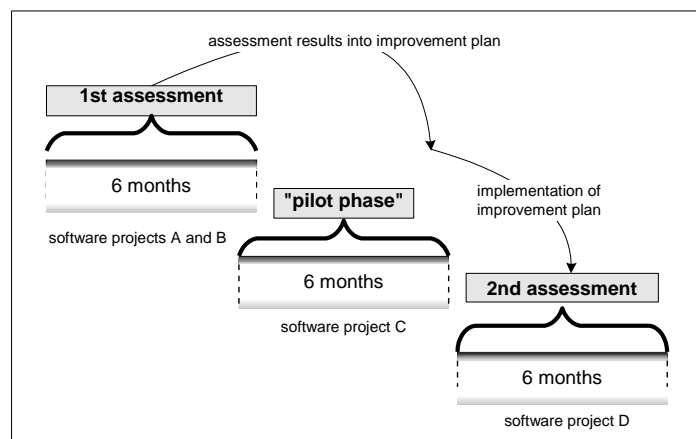


Fig. LEPASAAR.3 : The assessment-improvement scheme of a small software organisation [7]

During the first phase of the two software projects, named A and B, selected processes of the company are being assessed. The output of this phase is the assessment report describing the assessment results. During the software project C, the assessment results are analysed and an improvement plan drawn according to them. This phase represents also a “pilot phase” - employees usually draw their improvement ideas from the first assessment and try to implement them into practice. During the software project D the improvement plan is implemented. The second assessment takes place during the project D and it should already show some improved assessment results.

It is important to keep in mind that the first improvement takes place in the processes that are easiest to fix. Substantial improvement in processes takes far more than eighteen months. At the same time, constant work and devoted resource to software process improvement could also shorten the time period from the first assessment to the implementation of improvement plan. [7]

Conclusion

In this paper we have introduced the central part of the SPI environment – the SPI repository, which holds necessary information for various SPI related activities. The SPI environment aims to assist small software organisations in rapid and effective SPI activities. The small software organisations should receive a tool, which guides them through software process improvement, considering their organisations' characteristics, size and business goals, and diminishing the need for continuous external consultations and assessors.

The SPI repository is an organised data storage for supporting the software process improvement of small software organisations. It provides information about various SPI models and standards; best practices for benchmarking; assessment and improvement forms and templates to ease the SPI work; instructions of the usage of the environment; and examples and case-studies. In this article we have listed the content requirements for the SPI repository and following these requirements, we also give an example of the contents based on the SPICE model.

In future, we will specify the relationships of data provided in the RaSPI repository. We will also describe the application of the repository in greater detail to clarify the various possibilities for the usage of the RaSPI environment.

References

- [1] British Computer Society SPIN (UK)
<http://www.sys.uea.ac.uk/Research/researchareas/spi/objectiv.htm>
- [2] Humphrey W. S. Managing the Software Process, *The SEI Series in Software Engineering*, Software Engineering Institute, Addison-Wesley, 1989.
- [3] ISO/IEC TR 15504-5:1998 Information technology - Software process assessment - Part 5: An assessment model and indicator guidance.
- [4] ISO/IEC TR 15504-7:1998 Information technology - Software process assessment - Part 7: Guide for use in process improvement.
- [5] ISO/IEC TR 15504-9:1998 Information technology - Software process assessment - Part 9: Vocabulary.
- [6] The SPIRE Project Team. The SPIRE Handbook – Better, Faster, Cheaper Software Development in Small Organisation, *The European Community*, Centre of Software Engineering, 1998.
- [7] Lepasaar M. Software Process Improvement in Small Software Organisations – thesis for the degree of Master of Science, Tallinn, 2001.
- [8] Mäkinen T., Varkoi T. & Lepasaar M. A Detailed Process Assessment Method For Software SMEs, *Proceedings of the European Software Process Improvement (EuroSPI 2000) conference*, Copenhagen, 2000.
- [9] Varkoi T., Mäkinen T. & Jaakkola H. Process Improvement Priorities in Small Software Companies, *Proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET 99)*, Portland, Oregon, 1999 (CD-ROM).
- [10] Varkoi T., Mäkinen T. & Jaakkola H. Requirements for a Software Process Improvement Support and Learning Environment, *Proceedings of the Portland International Conference on Management of Engineering and Technology PICMET'01*, Portland, Oregon, 2001 (in press).
- [11] Zahran S. Software Process Improvement: Practical Guidelines for Business Success, *The SEI Series in Software Engineering*, Software Engineering Institute, Addison-Wesley, 1998.

About the authors

M.Sc. Marion Lepasaar - Ms. Lepasaar graduated MSc from Tallinn Technical University in 2001 and is currently working as a researcher for Pori School of Technology and Economics. She has studied SPI of small software organisations and compared SPIN activities in Finland and Estonia. Her research interests focus on Software Process Improvement and Assessment including SPICE, CMMI and technologies like UML.

Lic. Tech. Timo Varkoi - Mr. Varkoi has studied Computer Science at Tampere University of Technology and graduated Master of Science in 1989 and Licentiate of Technology in 2000. His working experience includes both software development and management of industrial software organizations. Recently he has been working as a project manager in an ESF-funded project to improve software processes with small software companies. He is a competent SPICE assessor and chairman of the SPICE-section of the Finnish Software Metrics Association (FiSMA). His current interests include software process assessments; software process improvement related research, consultation and training; development of software intensive organizations using EFQM and Balanced Scorecard.

Session 8 - SPI and Management

**Session Chair:
John Elliott, QinetIQ, UK**

Magical Management Commitment	8-2
Software Metrics: Uncertainty and Causal Modeling	8-8

Magical Management Commitment

Morten Korsaa
Continuous Improvement, Denmark

Introduction

Have you ever met a burned out SPI dude?

Have you ever seen an organisation that spent a lot of money and effort on SPI programs, but still did not improve significantly?

Have you ever met a manager who was frustrated over the delivery from the SPI department?

If you have, and would like to know why and maybe make a difference, then consider this:

On SPI conferences you find three types of presenters and attendants.

- *The consultants that offers all different kind of assistance.*
- *The top managers who explain how they changed their organisation*
- *The SPI people who often sigh: "Yes I could do that, if only I had the management commitment."*

Why are the top managers making it sound so easy, when the SPI folks are struggling so hard?

Well, SPI work is about techniques for e.g. Configuration Management and Requirement Management. But when it comes to deployment of the techniques, SPI becomes a matter of changing peoples and organisations behaviour. This paper will discuss what I find to be the root cause of failed SPI projects.

First I will give you the story of what normally happens!

What normally happens

Once upon a time in the industrial world in the beginning of the Web period, there was a bright programmer with strange sense of seeing how his team's daily work could be more efficient, and even produce better quality. He went to his manager and said: "If only we did XX before YY, things would be more efficient. His manager listened, and realised that he maybe was right, and then the manager became slightly embarrassed since he should have recognised that in the first place. He buried his embarrassment in an excuse by saying that he did not have time for that right now. Our friend went sadly away, but he just could not help thinking this way. After he have been preaching for some time his manager finally gets a bright idea, all by himself and say's: "Why don't you solve our problems my friend. Now you have a Software Process Improvement program, you are the process manager, God bless you, save our world". Our friend gets very excited over the thrust that management is showing him and hurries back to his desk, where he starts planning.

To know where you want to go, you have to know where you are, and our friend raises commitment for a process assessment (CMM/Bootstrap/SPICE/...). That was not an easy task, but the result is good. Or should we say it is bad, because it states that the organisation is not performing very well, but it states it very confident, based on the organisations own knowledge. After the assessment everybody knows where the problems are and our friend is expecting a lot of attention to his SPI program. He is really looking forward to help the projects with their problems. He is waiting for them to show some commitment. And he is waiting. And waiting... The projects are very busy; the next release is right around the corner, so they have no time now. But they would really like him to do all process improvement work, and then on their next project they promise to do it all.

Well, since our friend have the skills, and the dedicated time, eventually it seems like the best possible idea to start writing procedures. And he writes great procedures. Actually the best in the world... for him! If he were supposed to go out there and do the job, it would have been great processes. Really! But he is not, and since the projects where not involved, the processes does not reflect their reality enough to make them recognise that this is really solving their problems. Only if they change their practices, the business will improve. They don't, because the reward for engaging in the new processes was not obvious enough. So the great effort is ending up as yet another binder in the shelf.

If you recognise this story from somewhere near you, you are not the only one. From my own experience I would say that most of the attendants to a SPI conference are in the danger zone! For one simple reason, they are SPI people, and not operational managers. To explain that we have to take a short detour, hang on.

What should happen

Power and responsibilities

Who has the responsibility for the overall quality and efficiency of the organisation's software development?

Who has the responsibility for a given projects quality and efficiency?

Who has the power to change the salary for an employee that has performed extraordinary good (or bad)?

Who has the power to change the deadline for a project to allow them to study and exercise a smart estimation method?

Who has the power to give a 10% bonus to a project that documents less than 30% rework?

I assume that your answer to the questions above was either a project manager or a line manager. Did you answer "quality manager" or "SPI manager" to any of the questions? Probably not, and that is the reason why the quality or SPI manager never can have the responsibility for changing an organisation. He simply does not have any of the power it takes to manage. That is not bad by the way, he should not have. But very often the organisation expects the quality manager to be responsible for the quality. It lies in the name! The logistics manager is in charge of logistics and so on, but the quality manager can never be in charge of the quality because he is not empowered with the authority it would take. If management expects the quality manager to manage the quality, and the company still has a quality problem, then the quality manager most have a problem or be the problem himself. Precious time is wasted with the wrong focus. Please recognise this death spiral. This has been the root cause to many frustrations and burned out process colleagues.

There is one way to break this spiral of bad habits. Build the right environment in which the responsibilities are shared as described in the following paragraphs:

Line Management

The line management (every management level above project management) is responsible for the organisation. Including quality and efficiency. So be it. They hire and fire. They have the power to reward and punish according to their preference. What they find is important, points out the direction that the organisation will go. If that preference is based on quality and efficiency, they will influence in that direction. If it is not, it never will.

Delegating the responsibility for quality and efficiency is not an option. It remains forever the responsibility of management.

In practice this means that Line management must set the quality targets for the projects. Part of their strategy work is to identify the areas that need to improve to gain market advantage. Then they must set clear targets for these areas and measure the result. Finally they must reward the good results.

Project management

The project manager is responsible for the performance of his project. He will fill out the frames that are given by the line management. The frames are twofold:

Deliver the expected functionality.

Meet some quality targets

It may seem contradicting. (Most likely it is not however. In the big picture, it is not quality that costs. Bad quality is on the other hand extremely expensive. So even though it may not seem obvious, there is a pretty good chance that any given development towards better quality will pay off.) The important thing however is the project manager's view. If he believes that he can't meet both targets, guess which one he prioritises. The one that the line-managers rewards the best!

Anyway, the project manager's responsibility is to meet the quality targets set by the line management.

Quality support

The quality manager/support or SPI group has one primary purpose. Help the projects to meet the quality targets set by the line management. They may also help the line management to specify the quality targets. But the targets must never be communicated through their mouth, but always from line management. Note that quality managers/SPI managers/ Process support people are never responsible for quality and efficiency.

A different set up

Now a new situation builds up. The project is realising that they can not meet the quality targets, and ask quality support for help. And guess what! They are more than happy to assist, and they have the skills as well. The project manager will make a strategy to meet the targets, and develop the team's competencies accordingly with help from the quality support. Then the line management are happy to see that there their organisation is developing its competencies and business is improved.

If the responsibilities are shared as described above, the set up has changed dramatically. From being an annoying source of unpleasant changes the quality support is now a resource of insight and help towards a common goal. That makes all the difference if you want to change an organisation.

What should you do to change the situation?

Recognise your situation

*Check out your own environment. Who sets targets for quality in your organisation?
Targets like*

- *Rework*
- *Predictability*
- *Efficiency*
- *Failure rate*
- *Defects*
- *Deadlines*

What happens if those targets are not met? Does it affect salary and career opportunities.

Does management set those targets? Does Quality support? Projects? None at all?

Make a personal strategy

If you are in a position as a SPI/Quality/Process person, and find that your achievements hardly reward your effort, it is time to make a change. As discussed earlier, there can be many reasons for that, and it may not be due to your qualifications! If the framework you are working in is wrong, you cannot succeed no matter how good professional skills you have. And it is wrong if you are known to be responsible for quality/efficiency and you are not in a position to set targets and reward. There are two obvious strategies:

Take control - real control.

Move to management! Maybe you should have been a manager! You have at least some skills that are needed! You have the process optimising mindset. You have the quality mindset. But when you plan your "career move" these are not your selling points. Keep in mind, that if those mindset were valued in your organisation the problem would not have been there in the first place. You would all ready have been a manager or the position you are in now would never have been needed! Make a career strategy based on what is valued, and sneak in all your good stuff through the back door. In this strategy you may consider if changing to another company is an opportunity. The label you get in a workplace can be tough to change. It may be easier to change the workplace.

Give away control!

Drop the management responsibilities. Identify all the daily situations where you are expected to be responsible for something that really is the management's responsibilities. Then make it clear that it is not your problem, but you would be very happy to help those who have the problem if only they ask. E.g. you are "Quality manager" and the CEO calls you and say that there is a problem with the latest

product, and ask what you plan to do about it. Your first reply is that you don't expect to do anything. You are not producing the product so how can you change the quality of the product. Then you offer your service if the CEO wants to know what he can do, or if the project wants to know what they can do to change the quality of the product. That creates the right framework for all parties.

Conclusion

The reason for the top managers to be present at the conferences is to share how they changed their organisation. Why can they do that? Because they are managers. They have the power it takes. And they have the software process skills or at least a good SPI person that support them. But they drive the organisational change process. And that is the only magic. It makes a big difference who is giving the message, even though the logic of the words are the same. Try this sentence:

Last year we had a rework ratio of 40 % according to our definitions. Next year I want this figure to be 25%. You can reach this figure anyway you want, but let me suggest that you take a look at the correctness of your requirements. Projects who reach this target will receive a 10% extra bonus, and if all projects do so, an additional 10% company bonus."

Imagine it said by a quality support person. Then imagine that the CEO says it at the yearly strategy day. It makes all the difference!

The day you either hear your manager say these words, or you say them yourself in your new management role, then you recognise the magic management commitment. Then will you find less burned out SPI people, and the business will improve.

Software Metrics: Uncertainty and Causal Modelling

Norman Fenton, Martin Neil
*Agena Ltd, 11 Main Street, Caldecote,
Cambridge, CB3 7NU*

Paul Krause
*Philips Research Laboratories,
Crossoak Lane, Redhill, Surrey, RH1 5HA*

Roger Shaw
*Queen Mary, University of London,
Mile End Road, London, E1 4NS*

Introduction

Important decisions need to be made during the course of developing software products. Perhaps the most important of these is the decision when to release the software product. The consequences of making an ill-judged decision can be potentially critical for the reputation of a product or its supplier. Yet, such decisions are often made informally, often when time and budget constraints are approached, rather than on the basis of more objective and accountable criteria.

Software project and quality managers must juggle a combination of uncertain factors, such as tools, personnel, development methods and testing strategies to achieve the delivery of a quality product to budget and on time. Each of these uncertain factors influences the introduction, detection and correction of defects at all stages in the development life cycle from initial requirements to product delivery.

In order to achieve software quality during development special emphasis needs to be applied to the following three activities:

- Defect prevention: If the number of defects introduced during specification, design and coding stages can be reduced, confidence in delivered quality can be enhanced. Applying ‘formal’ methods for system design and requirements can help reduce the ambiguities and inconsistencies that might lead to significant defects.
- Defect detection: Testing program parts (modules) and the final system, before delivery, will result in early defect detection and provide a measurement of program quality. Manual review and inspection of design and specification documentation can also contribute to the overall quality of the delivered product.
- Defect correction: If defects have been discovered, during testing, it is important to ensure that they are fixed correctly. However because software is extremely complex fault diagnosis can be difficult, and may result in the introduction of additional faults.

The challenge of software development is to apply finite resources to all of these activities and, based on the division of resources applied, predict the likely quality that will be achieved. To date the majority of software projects have tended to rely upon the judgement of the project or quality manager. Unfortunately, where mathematical or statistical procedures have been applied, their contribution has been marginal at best [1].

Our aim here is to produce a single model allowing the combination of (often causal) diverse evidence in a more natural and efficient way than done previously. We use graphical probability models as the appropriate formalism for representing this evidence. We can use the subjective judgements of experienced project managers to build the probability model and use this model to forecast software quality throughout the development life cycle. Moreover, the causal or influence structure of the model provides a more natural depiction of the world of software development

The paper starts by outlining some of the current approaches to quality modelling. Such models should be able to guide project management decisions. Next we identify some of the biases that can effect decision-makers in the absence of a sound decision making framework. We review some of the difficulties associated with current approaches to quality prediction. Our approach draws on the use of probability to enable techniques from classical decision theory to be used in guiding software release and process improvement decisions. Since these are relatively new areas for most software engineers, we provide an introduction to the method used in this paper.

Current approaches to software quality modelling

In this section we will look at the general issues relating to quality control and assessment in software development. In subsequent sections attention will be given to defect modelling. However, it is worth phrasing the problem in general terms to emphasise that the longer-term goal is to apply probabilistic graphical models to other quality characteristics, like reliability and safety [2, 3].

There are two different viewpoints of software quality as defined by Fenton and Pfleeger [4]. The first, the external product view, looks at the characteristics that make up the user’s perception of quality in the final product – this is often called quality-in-use.

Quality-in-use is determined by measuring external properties of the software, which can only be secured once the software product is complete. For instance quality here might be defined as freedom from defects or the probability of executing the product, failure free, for a defined period.

The second viewpoint, the internal product view, involves criteria that can be used to control the quality of the software as it is being produced and that can form early predictors of external product quality. Good development processes and well-qualified staff working on a defined specification are just some of the pre-requisites for producing a defect free product. If we can ensure that the process conditions are right, and can check intermediate products to ensure this is so, then we can perhaps produce high quality products in a repeatable fashion.

Unfortunately the relationship between the quality of the development processes and the resulting quality of the end products is not deterministic. Software development is a profoundly intellectual and creative design activity with vast scope for error and for differences in interpretation and understanding of requirements. The application of even seemingly straightforward rules and procedures can result in highly variable practices by individual software developers. Under these circumstances the relationships between internal and external quality are uncertain.

Typically informal assessments of critical factors will be used during software development to assess whether the end product is likely to meet requirements:

- Complexity measures: A complex product may indicate problems in the understanding of the actual problem being solved. It may also show that the product is too complex to be easily understood, de-bugged and maintained.
- Process maturity: Development processes that are chaotic and rely on the heroic efforts of individuals can be said to lack maturity and will be less likely to produce quality products, repeatedly.
- Test results: Testing products against the original requirements can give some indication of whether they are defective or not. However, the results of testing are only as good as the testing process itself.

Evidence as described is often collected in a piecemeal fashion and used to inform management about the quality of the final product. However there is often no formal attempt, in practice, to combine these different sources of evidence into a single quality model.

Many 'models' of software quality have been defined. Figure FNK1 is typical of the earliest models produced by McCall, Boehm and others [5]. Firstly it provides a graphical representation of the relations between factors and the criteria one could use to 'measure' or 'indicate' those factors. This type of structure seems suggestive of a graphical probability model but the relations are too unclear and imprecise to be represented by probabilities, despite the apparent uncertainty in the relations.

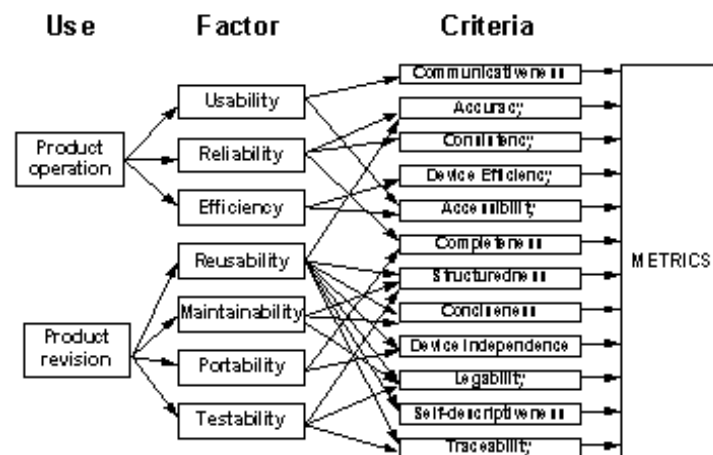


Figure FNK1: Quality model based on the work of Boehm and McCall.

In this paper we develop a qualitative model, but one with clearer semantics and some predictive and explanatory structure. This structure involves combining the internal quality variables with measures of the external quality variables (defects) throughout the development process. However, its validity has not yet been subject to controlled empirical studies, and there is no well-established and universal quantitative link between the development process and the quality of the delivered product. As a result, decisions using expert judgement are made on the basis of evidence taken from product and process attributes. Such judgements are likely to be subject to bias and influence from project pressures. In addition, these subjective judgements are very hard to defend if they are contrary to the wishes of other project stakeholders.

Judgmental bias in software development

Basic forms of bias

The issue of bias in human judgement is one of intense debate amongst cognitive psychologists. In this section we will illustrate a classic example of a form of bias that appears in decision making in software development, and then state some general requirements for a more effective decision making framework.

The forms of bias are generally classified under four headings [6]:

- conservatism bias;
- base-rate neglect;
- the gambler's fallacy;
- overconfidence bias.

Of these, the one we have seen most evidence of in software development is base-rate neglect: insufficient account being taken of the initial or background knowledge that is available. A simple example is evident in the development of successive releases of a software product: if the quality of the base product is poor, then this can dominate the quality of the up-grade no matter how carefully the latter is developed and tested. Yet we often see examples of heroic efforts being made to somehow improve the quality of the later releases, to little effect.

How much can you trust an imperfect test?

The significance of base-rate neglect can be made more tangible by drawing a direct analogy with the classic Harvard Medical School example [7]. The following question was asked to 60 students and staff in the Harvard Medical School.

Suppose a particular heart disease has a prevalence of 1 in 1000 people. This establishes the prior information, or base-rate. A test to detect this disease is available. The test is not perfect and has a false positive rate of 5%. However, the false negative rate is zero. That is, the test diagnoses correctly every person who has the disease. What is the chance that a randomly selected person found to have a positive result actually has the disease?

Of the 60 students and staff, half gave the answer 95%. 11 participants only gave the correct answer. The answer of 95% takes no account of the base-rate for the heart disease. Indeed this answer also indicates a lack of understanding of what is meant by the false positive rate. There are several ways to obtain the correct answer, but perhaps the most intuitive is as follows.

Consider 10,000 people chosen at random. From the base rate, 10 will suffer from the heart disease. These will give a positive result when tested. Because of the false positive rate, 500 will erroneously give a false positive test. So, of those giving positive tests only 10 out of 510 will actually have the heart disease. That is, approximately 2%. In this particular example, the low base-rate heavily dominates the conclusions that can be drawn from an imperfect test. The relatively low chance that someone who tests positive actually suffers from the disease is critical.

We can draw a direct analogy with software quality estimates drawn from limited testing. Assuming that the tests are correctly specified, the false negative rate will be zero (if there are no faults in the product, the test suite will always be successfully exercised). However, unless a software product is tested exhaustively, there will always be a chance that a fault will go undetected. That is, a software test suite will have a false positive rate. So, unless we are 100% confident in the quality of a software product before it goes into test, the test suite will warrant relatively limited confidence in the quality of the software even if no failures are experienced as the test suite is exercised.

Frequently software products may be shipped following limited testing, during which no failures were observed. One might, at first glance, assume that zero failures detected equates to a quality product. However, the argument presented above makes it clear that

the confidence in the quality of the product may be unjustified unless one could be assured independently that:

- the product is actually defect free, explaining why testing found no defects; or
- testing was extremely rigorous and could be trusted to give a highly accurate test result.

Unfortunately, software developers can suffer from the erroneous belief that, based perhaps on data from the development process alone, the product is of high quality and is one that perhaps does not need expensive testing – if it is correct why test it?

Correct judgement of the confidence in the quality of the software must take into account both a prior judgement in the quality of the software given the development processes used, and an assessment of the false positive rate of the test suite. Note that the preceding example is quite extreme because of the low base-rate of the hypothetical heart disease. But the lesson remains that successful execution of a test suite may not warrant as high a degree of confidence in the quality of a software product as one thinks.

Requirements for decision support in software development

In order to improve the situation we need a model that clearly:

1. links controllable quality criteria (development methods, staff skills, and so forth) to the quality-in-use of the final implemented software;
2. adequately models the inaccuracy of software testing as a measure of software quality;
3. does both of the above over a complex development life-cycle;
4. can guide and improve the development process, or at least educate managers about the importance of fault prevention and detection.

Guidance from a “gold standard” framework can provide significant support to enable the “optimal” judgements to be made, given the information that is available. This provides the basic motivation for investigating whether the recent developments in probabilistic graphical modelling can be used to provide support for more effective decision making in software engineering. One goal of the remainder of this report is to illustrate how a probabilistic model can meet the above requirements. However, we should first state why we think existing quality prediction models are not adequate. This will be done in the next section.

The problems with software quality prediction

Very often a software release decision is made on the basis of the expert judgement of an individual or group of individuals. A number of models are available which can be used for software quality prediction and these could be used to provide additional support for the software release decision. Unfortunately, a number of problems are associated with these models.

The use of size or complexity metrics as the sole predictors of defects

Size and complexity metrics are available during product development, and can in principal be used as ‘controls’ for product quality. For example, some development organisations adopt coding conventions that constrain software modules to have less than a specified number of lines of code, or have a complexity figure below a certain threshold. There are certain pragmatic difficulties with this (for example, a perfectly respectable switch statement construct for a user interface function may have a high complexity figure, yet offer no inherent threat to product quality). However, the main concern is that the causal relation between size or complexity and any of the characteristics of quality in use is not established.

We can illustrate this with some contrasting published experiences. Studies that indicate a positive correlation between defect density (dependent variable) and program size (independent variable) have been published by Akiyama [8], Ferdinand [9], Halstead [10] and Ottenstein [11]. In contrast, Hamer and Frewin [12], Shen et al. [13] and Shepperd [14] have collected empirical evidence that refutes the validity of defect prediction models based on size and complexity.

We can begin to get a handle on this apparently contradictory state of affairs, by making two observations:

- Firstly, the measures of “complexity” used are better interpreted as measures of “size”, so we can concentrate on “size” as the relevant software attribute.
- Secondly, we model the situation by hypothesising “problem difficulty” and “design ability” as common causes of program size and the number of defects introduced. This means that depending on how the experiment is set up (intervening with the common causes), different results will be obtained; the discrepancies are due to incompletely specified experiments.

Problems with multivariate approaches

Applying multivariate techniques, like factor analysis, produces metrics which cannot be easily or directly interpretable in terms of program features. For example, in [15] a factor dimension metric, control, was calculated by the weighted sum:

$$control = a_1 HNK + a_2 PRC + a_3 E + a_4 VG + a_5 MMC + a_6 Error + a_7 HNP + a_8 LOC$$

The a_i 's are derived from factor analysis. HNK was Henry and Kafura's information flow complexity metric, PRC is a count of the number of procedures, E is Halstead's effort metric, VG is McCabe's complexity metric, MMC is Harrison's complexity metric and LOC is lines of code. Although this equation might help to avoid multicollinearity it is hard to see how you might advise a programmer or designer on how to re-design the programs to achieve a “better” control metric value for a given module. Likewise the effects of such changes in module control on defects is less than clear.

The fundamental difficulty is that once again, one is not producing anything like a causal model of the domain. The formula for control, above, is not identifying differing, well-defined attributes with single standard measures. Instead it is taking weighted averages from several measures of essentially the same attributes; size and (possibly)

complexity. This is rather like proposing a weighted average of Fahrenheit and Centigrade as a more effective measure of temperature than either scale on its own. In truth, this approach manifestly does not address the fundamental weakness of directly relating size and complexity measures to defect densities that was identified in the previous section.

The “Goldilock’s Conjecture”

We can crystallise many of the above points by analysing the debate that surrounds the so call Goldilock’s Conjecture. This is a conjecture that there is an optimum module size that is “not too big nor too small”. Supporting this is Hatton’s [16] claim that there is “compelling empirical evidence from disparate sources to suggest that in any software system, larger components are proportionally more reliable than smaller components”.

If these results were generally true the implications for software engineering would be very serious indeed. It would mean that program decomposition as a way of solving problems simply did not work. Virtually all of the work done in software engineering extending from fundamental concepts, like modularity and information-hiding, to methods like object-oriented and structured design would be suspect because all of them rely on some notion of decomposition. If decomposition doesn’t work then there would be no good reason for doing it.

A detailed critique of the empirical evidence that is claimed to substantiate the Goldilock’s Conjecture can be found in [1]. What is of interest here are the specific problems that were found with the various studies:

- Imprecisely defined concepts. For example, none of the studies defined ‘module’ in such a way as to make comparison across data sets possible.
- Lack of account of software development process variables. None of the studies explicitly compared different approaches to structuring and decomposing designs.
- Problems with data quality and statistical methodology – the data analysis or quality of the data could not support the results claimed.
- Missing explanatory or causal variables – a number of factors exist that could partly explain the results which these studies have neglected to examine.

The basic problem is that curve fitting is too simplistic to handle the complex relationship between module size and defect density. The above analysis suggests that the following requirements must be satisfied if we are to enable effective and generalisable reliability prediction models to be built:

- The variables in the model must correspond to clearly defined attributes having standard measures associated with them.
- The model must incorporate variables corresponding to process attributes as well as product attributes.
- Associations between variables in the model must correspond to meaningful and testable causal relationships.
- The model must clearly distinguish between fault densities and failure rates, with the latter being the reliability attribute that is of interest to the end-user.

We believe that graphical probabilistic models are currently the best candidates for satisfying all the above requirements.

Introduction to probabilistic models

When I say 'S is probably P', I commit myself guardedly, tentatively or with reservations to the view that S is P, and (likewise guardedly) lend my authority to that view.

S. Toulmin, 1958

How, then, should I decide?

We live in an uncertain world. Rarely can we predict the outcome of a course of action with certainty. Yet, in order to plan and to act we must often choose from between two or more competing alternatives. The question then arises, how should one choose?

Classical decision theory is unequivocal on this point. Firstly, one identifies the possible outcomes of the various actions that are open in a given situation. Each outcome will have an associated utility (or perhaps disutility) – a figure expressing the financial gain (or loss) associated with the outcome. Then, and this is often the hard part, for each possible action one assesses the probability of the various outcomes occurring. Multiplying this probability by the utility of the respective outcome will then yield a measure of expected utility for each action. According to classical decision theory, one then simply chooses that action which maximises the expected utility.

Let's take a simple example to clarify this. Suppose one is offered a choice of two bets:

- Bet 1 has a 90% chance of winning £10.00
- Bet 2 has a 10% chance of winning £100.00

Neither has an associated cost. What action should you favour? We can go through this step by step:

1. What actions could you take?

There are two actions to decide between (ignoring the possibility of doing nothing, with no possible reward): accept Bet 1, or accept Bet 2.

2. What are the possible outcomes of those actions?

There are two possible outcomes for each action: either win the bet or lose the bet. In the case of winning Bet 1, the outcome will be a gain of £10.00. In the case of winning Bet 2, the outcome will be a gain of £100.00. In the case of losing either bet, there will be neither any gain, nor any loss.

3. What are the chances of those outcomes arising?

We are told there is a probability of 0.9 that we will win Bet 1, and a probability of 0.1 that we will win Bet 2.

4. Multiply the value of each outcome by the probability of it occurring.
This yields an “expected utility” of $0.9 \times £10.00 = £9.00$ for Bet 1, and $0.1 \times £100.00 = £10.00$ for Bet 2.
5. Choose that action which maximises the expected utility.

In this case we should choose to accept Bet 2.

It is important to realise that there are no guarantees of success – one could still lose this specific bet. However, in a sequence of such choices there is no other strategy that will yield a higher rate of success.

The procedure is well founded and apparently quite simple. Unfortunately, it is not so easy to follow in practice. In most real-world situations, it is very hard to assess the probabilities of the various outcomes. In some cases, even attaching financial values to the possible outcomes is hard and controversial (the value of a person’s life, in a decision involving risk of loss of life, being a classic example). However, there are techniques that can be used to help assess the probabilities. The issues associated with utilities will not be addressed further in this paper (although [17] can be referred to for some good examples).

The basics of probability theory

The basics of probability theory have been developed over some four hundred years into a highly sophisticated theory. Yet the principles behind it remain quite simple.

When using probabilities, one may talk in terms of, for example: the probability that a cancer patient will respond to a certain form of chemotherapy; the probability that a projectile might hit a region of space; the probability of observing a string of three identical outcomes in six dice throws. Probabilities conform to three basic axioms:

- $p(A)$, the probability of an event A (outcome/consequence...), is a number between 0 and 1;
- $p(A)=0$ means A is impossible, $p(A)=1$ means A is certain;
- $p(A \text{ or } B) = p(A) + p(B)$ provided A and B are disjoint.

Sometimes we can estimate the probabilities we want by counting (the ratio of the number of cancer patients cured to the total number treated, for example) or some other form of direct measurement. In this sense, we are saying that the probability is an attribute or property of the real world. The term *physical probability* is often used to denote this interpretation of probability.

Of course, any act of measurement has an element of imprecision associated with it. So, we would expect the probabilities obtained by measurement also to be imprecise; strictly any physical probability should be represented by a distribution of possible values. In general, the more information we have, the tighter will be the distribution. Sometimes, however, we will have no direct physical measurements by which to estimate a probability. An example of such a case might be when one is asked to toss a coin one has never seen before and judge the probability that it will land heads up. If one believes

the coin to be fair, an estimate of 0.5 for this physical probability would seem reasonable. For more complex situations, the value elicited for such a probability may vary from subject to subject: perhaps dependent on the level or relevance of expertise of the subjects.

When reasoning under uncertainty, one often needs to elicit the probability of an event for which there is no historical data. The case of the “virgin” coin in the previous paragraph is an example. In the absence of data on the frequency of “heads” events for this specific coin, we use an expert to judge the probability of this event. This is an example of a *subjective* probability, and its value will typically be conditional on the knowledge of the expert (an expert who has only ever seen double-sided heads coins may give a subjective probability of 1.0 for the same event).

As an aside, we make two further observations on subjective and physical probabilities. Firstly, experts are generally “well-calibrated” when asked to provide probabilities from within well-defined problem areas. This means there is a high degree of inter-subject consistency for subjective probabilities in these cases [6]. Secondly, subjective probabilities can be revised in a coherent way using subsequent experience of the real-world events of interest (e.g. [18]). These observations support a view of probability as a *measure* of propensity, with subjective probabilities as an estimate of that measure which can be increased in precision by the use of physical (“real-world”) data [19].

Conditional probability

Merely to refer to the probability $p(H)$ of an event or hypothesis is an oversimplification. In general, probabilities are context sensitive. For example, the probability of suffering from certain forms of cancer is higher in Europe than it is in Asia. Strictly, the probability of any event or hypothesis is conditional on the available evidence or current context. This can be made explicit by the notation $p(H | E)$, which is read as “the probability of H given the evidence E”. In the coin example, H would be a “heads” event and E an explicit reference to the evidence that the coin is a fair one. If there was evidence E' that the coin was double sided heads, then we would have $p(H | E') = 1.0$.

As soon as we start thinking in terms of conditional probabilities, we begin to need to think about the structure of problems as well as the assignment of numbers. To say that the probability of an hypothesis is conditional on one or more items is to identify the information relevant to the problem at hand. To say that the identification of an item of evidence influences the probability of an hypothesis being valid is to place a directionality on the links between evidences and hypotheses.

Often a direction corresponding to causal influence can be the most meaningful. For example, in medical diagnosis one can in a certain sense say that measles “causes” red spots (there might be other causes). So, as well as assigning a value to the conditional $p(\text{'red spots'} | \text{measles})$, one might also wish to provide an explicit graphical representation of the problem. In this case it is very simple (Figure FNK2).

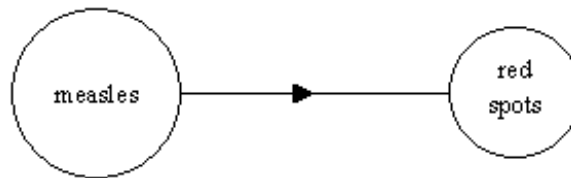


Figure FNK2: A very simple probabilistic network

Note that to say that $p(\text{'red spots'} \mid \text{measles}) = p$ means that we can assign probability p to 'red spots' if measles is observed and only measles is observed. If any further evidence E is observed, then we will be required to determine $p(\text{'red spots'} \mid \text{measles}, E)$. The comma inside the parentheses denotes conjunction.

Building up a graphical representation can aid framing the problem. A significant recent advance in probability theory [21] has been the demonstration of a formal equivalence between the structure of a graphical model and the dependencies that are expressed by a numerical probability distribution. In numerical terms, we say that event A is independent of event B if observation of B makes no difference to the probability that A will occur: $p(A \mid B) = p(A)$. In graphical terms we indicate that A is independent of B by the absence of any direct arrow between the nodes representing A and B .

So far, we have concentrated on the static aspects of assessing probabilities and indicating influences. However, probability is a dynamic theory; it provides a mechanism for coherently revising the probabilities of events as evidence becomes available. Conditional probability and Bayes' Theorem play a central role in this. We will not cover the details of Bayesian updating here, but will use a simple example to illustrate the qualitative behaviour instead.

Suppose we are interested in the number of defects that are detected and fixed in a certain testing phase. If the software under test had been developed to high standards, perhaps undergoing formal reviews before release to the test phase, then the high quality of the software consequent upon using formal reviews or similar would in a sense "cause" a low number of defects to be detected in the test phase. However, if the testing were ineffective and superficial, then this would provide an alternative cause for a low number of defects being detected during the test phase.

This situation can be represented by the simple graphical model of Figure FNK3. Here the nodes in the graph could represent simple binary variables with states "low" and "high". However, in general a node may have many alternative states or even represent a continuous variable. We will stay with the binary states for ease of discussion.

It can be helpful to think of Figure FNK3 as a fragment of a much larger model. In particular, the node SQ ("Software Quality") could be a synthesis of, for example:

- review effectiveness;
- developer's skill level;
- quality of input specifications;
- resource availability.

With appropriate probability assignments to this model, a variety of reasoning styles can be modelled. A straightforward reasoning from cause to effect is possible. If TE (test effectiveness) is “low”, then the model will predict that DD (defects discovered and fixed) will also be low. If earlier evidence indicates SQ (software quality) is “high”, then again DD will be “low”.

However, an important feature is that although conditional probabilities may have been assessed in terms of effect given cause, Bayes’ rule enables inference to be performed in the “reverse” direction – to provide the probabilities of potential causes given the observation of some effect. In this case, if DD is observed to be “low” the model will tell us that low test effectiveness or high software quality are possible explanations (perhaps with an indication as to which one is the most likely explanation). The concept of “explaining away” will also be modelled. For example, if we also have independent evidence that the software quality was indeed high, then this will provide sufficient explanation of the observed value for DD and the probability that test effectiveness was low will be reduced.

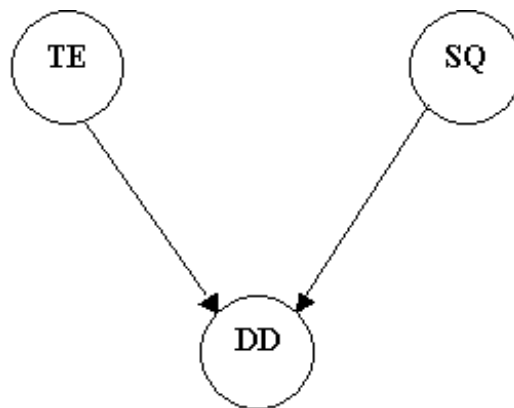


Figure FNK3: Some subtle interactions between variables captured in a simple graphical model. Node TE represents “Test Effectiveness”, SQ represents “Software Quality” and DD represents “Defects Detected and Fixed”.

This situation can be more formally summarised as follows. If we have no knowledge of the state DD then nodes TE and SQ are marginally independent – knowledge of the state of one will not influence the probability of the other being in any of its possible states. However, nodes TE and SQ are conditionally dependent given DD – once the state of DD is known there is an influence (via DD) between TE and SQ as described above.

We will see in the next section that models of complex situations can be built up by composing together relatively simple local sub-models of the above kind (see also [24]). This is enormously valuable. Without being able to structure a problem in this way it can be virtually impossible to assess probability distributions over large numbers of variables. In addition, the computational problem of updating such a probability distribution given new evidence would be intractable.

Bayes’ theorem and graphical models

We should introduce Bayes' theorem before moving on to the structuring of complex problems using graphical models. As indicated in the previous section, probability is a dynamic theory; it provides a mechanism for coherently revising the probabilities of events as evidence becomes available. Bayes' theorem is a fundamental component of this dynamic dimension.

We write $p(A|B)$ to represent the probability of some event (an hypothesis) conditional on the occurrence of some event B (evidence). If we are counting sample events from some universe Ω , then we are interested in the fraction of events B for which A is also true. In effect we are focusing attention from the universe Ω to a restricted subset in which B holds. From this it should be clear that (with the comma denoting conjunction of events):

$$p(A|B) = \frac{p(A, B)}{p(B)}$$

This is the simplest form of Bayes' rule. However, it is more usually rewritten in a form that tells us how to obtain a posterior probability in a hypothesis A after observation of some evidence B, given the *prior* probability in A and the likelihood of observing B were A to be the case:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

This theorem is of immense practical importance. It means that we can reason both in a forward direction from causes to effects, and in a reverse direction (via Bayes' rule) from effects to possible causes. That is, both deductive and abductive modes of reasoning are possible.

Two significant problems need to be addressed. Although in principle we can use generalisations of Bayes' rule to update probability distributions over sets of variables, in practice:

- 1) Eliciting probability distributions over sets of variables is a major problem. For example, suppose we had a problem describable by seven variables each with two possible states. Then we will need to elicit (2^7-1) distinct values in order to be able to define the probability distribution completely. As can be seen, the problem of knowledge elicitation is intractable in the general case.
- 2) The computations required to update a probability distribution over a set of variables are similarly intractable in the general case.

Up until the late 1980's, these two problems were major obstacles to the rigorous use of probabilistic methods in computer based reasoning models. However, work initiated by Lauritzen and Spiegelhalter [20] and Pearl [21] provided a resolution to these problems for a wide class of problems. This work related the independence conditions described in graphical models to factorisations of the joint distributions over sets of variables. We have already seen some simple examples of such models in the previous section. In probabilistic terms, two variables X and Y are independent if $p(X, Y) =$

$p(X)p(Y)$ – the probability distribution over the two variables factorises into two independent distributions. This is expressed in a graphic by the *absence* of a direct arrow expressing influence between the two variables.

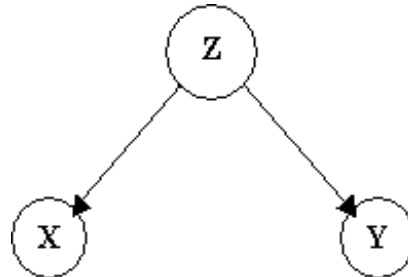


Figure FNK4: X is conditionally independent of Y given Z.

We could introduce a third variable Z, say, and state that “X is conditionally independent of Y given Z”. This is expressed graphically in Figure FNK4. An expression of this in terms of probability distributions is:

$$p(X, Y|Z) = p(X|Z)p(Y|Z)$$

A significant feature of the graphical structure of Figure FNK4 is that we can now decompose the joint probability distribution for the variables X, Y and Z into the product of terms involving at most two variables:

$$p(X, Y, Z) = p(X|Z)p(Y|Z)p(Z)$$

In a similar way, we can decompose the joint probability distribution for the variables associated with the nodes DD, TE and SQ of Figure FNK3 as

$$p(DD, TE, SQ) = p(DD|TE, SQ)p(TE)p(SQ)$$

This gives us a series of example cases where a graph has admitted a simple factorisation of the corresponding joint probability distribution. If the graph is directed (the arrows all have an associated direction) and there are no cycles in the graph, then this property is a general one. Such graphs are called Directed Acyclic Graphs (DAGs). Using a slightly imprecise notation for simplicity, we have [20]:

Proposition

Let $U = \{X_1, X_2, \dots, X_n\}$ have an associated DAG G . Then the joint probability distribution $p(U)$ admits a direct factorisation:

$$p(U) = \prod_{i=1}^n p(X_i | pa(X_i))$$

Here $pa(X_i)$ denotes a value assignment to the parents of X_i . (If an arrow in a graph is directed from A to B, then A is a parent node and B a child node).

The net result is that the probability distribution for a large set of variables may be represented by a product of the conditional probability relationships between small

clusters of semantically related propositions. Now, instead of needing to elicit a joint probability distribution over a set of complex events, the problem is broken down into the assessment of these conditional probabilities as parameters of the graphical representation.

The lessons from this section can be summarised quite succinctly. First, graphs may be used to represent qualitative influences in a domain. Secondly, the conditional independence statements implied by the graph can be used to factorise the associated probability distribution. This factorisation can then be exploited to (a) ease the problem eliciting the global probability distribution, and (b) allow the development of computationally efficient algorithms for updating probabilities on the receipt of evidence. We will now describe how these techniques have been exploited to produce a probabilistic model for software defect prediction.

The Probabilistic Model for Defect Prediction

We came up with a number of requirements for an effective model of software defect prediction at the end of the last section. Probabilistic models are a good candidate solution as:

1. They can easily model causal influences between variables in a specified domain;
2. The Bayesian approach enables statistical inference to be augmented by expert judgement in those areas of a problem domain where empirical data is sparse;
3. As a result of the above, it is possible to include variables in a software reliability model that correspond to process as well as product attributes;
4. Assigning probabilities to reliability predictions means that sound decision making approaches using classical decision theory can be supported.

Our goal was to build a module level reliability model which could then be evaluated against real project data. Resources were not available to perform extensive knowledge elicitation with the active and direct involvement of members of Philips' development organisations. Philips Research Labs' experience from working directly with the business units was used as a surrogate for this. This meant that the probabilistic network could be built within a relatively short period of time. However, the fact that the probability tables were in effect built from "rough" information sources and strengths of relations necessarily limits the precision of the model.

The remainder of this section will provide an overview of the model to indicate the product and process factors that are taken into account when a quality assessment is performed using it.

Overall structure of the probabilistic network

The probabilistic network was built using the generic probabilistic inference engine Hugin (see <http://www.hugin.com> for further details). However, the size and complexity of the network were such that it was not realistic to attempt to build the network directly using the Hugin tool. Instead, Agena Ltd used two methods and tools that have built on

top of the Hugin propagation engine:

- The SERENE method and tool [22] which enables: large networks to be built up from smaller ones in a modular fashion; and, large probability tables to be built using pre-defined mathematical functions and probability distributions.
- The IMPRESS method and tool [23], which extends the SERENE tool by enabling users to generate complex probability distributions simply by drawing distribution shapes in a visual editor.

The resulting network takes account of a range of product and process factors from the lifecycle of a software module. Because of the size of the model, it is impractical to display it in a single figure. Instead, we provide a first schematic view in terms of sub-nets. This modular structure is the actual decomposition that was used to build the network using the SERENE tool.

The main sub-nets in the high-level structure correspond to key software life-cycle phases in the development of a software module. Thus there are sub-nets representing the specification phase, the specification review phase, the design and coding phase and the various testing phases. Two further sub-nets cover the influence of requirements management on defect levels, and operational usage on defect discovery. The final Defect density sub-net simply computes the industry standard defect density metric in terms of residual defects delivered divided by module size.

This structure was developed using the software development processes from a number of Philips development units as models. A common software development process is not currently in place within Philips. Hence the resulting structure is necessarily an abstraction. Again, this will limit the precision of the resulting predictions. Work is in progress to develop tools to enable the structure to be customised to specific development processes.

The arc labels in Figure FNK5 represent ‘joined’ nodes in the underlying sub-nets. This means that information about the variables representing these joined nodes is passed directly between sub-nets. For example, the *specification quality* and the *defect density* sub-nets are joined by an arc labelled ‘Module size’. This node is common to both sub-nets. As a result, information about the module size arising from the specification quality sub-net is passed directly to the defect density sub-net. We refer to ‘Module size’ as an ‘output node’ for the *specification quality* sub-net, and an ‘input node’ for the *defect density* sub-net. The figures in the following sub-sections show details of a number of sub-nets. In these figures, the dark shaded nodes with dotted edges are output nodes, and the dark shaded ones with solid edges are input nodes (FNK6 to FNK10).

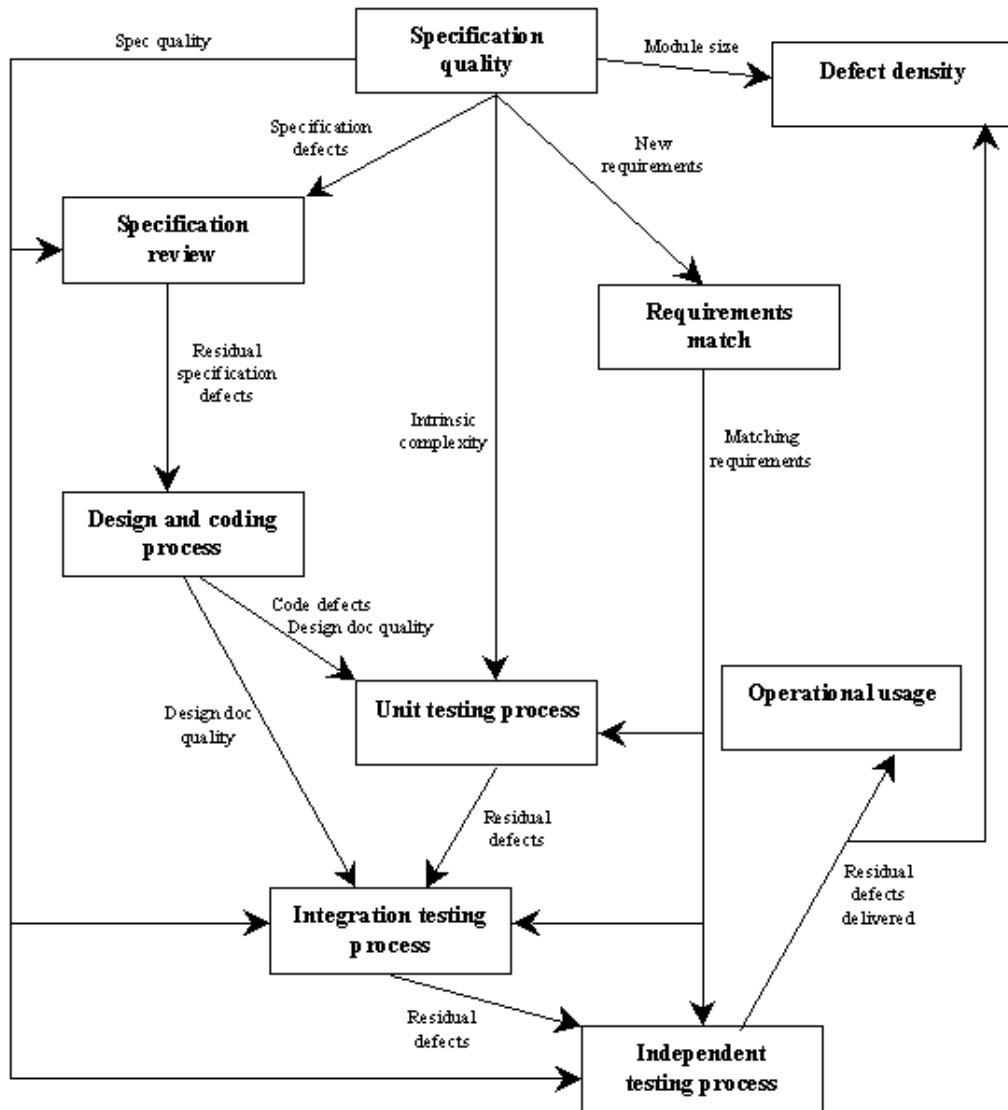


Figure FNK5: Overall network structure.

The specification quality sub-net

Figure FNK6 illustrates the Specification quality sub-net. This sub-net has a feature that is common to a number of the others. Specifically, it contains some intermediate nodes that are used purely to manage some computations. These nodes are usually the single parent of a node that has the postscript “effects”. Thus, for example the node labelled “stability” is one of these intermediate nodes since it is the parent of a node labelled “stability effects”. The node stability, which has many state values, is used to compute an appropriate combination of its own parent nodes (the factors that really affect stability). However, these values are then transformed into a sensible scale in the node “stability effects”. These intermediate nodes will not be of interest to a user of the BBN, and should ideally be hidden from the user; unfortunately, the current functionality of the Hugin tool does not allow us to do this.

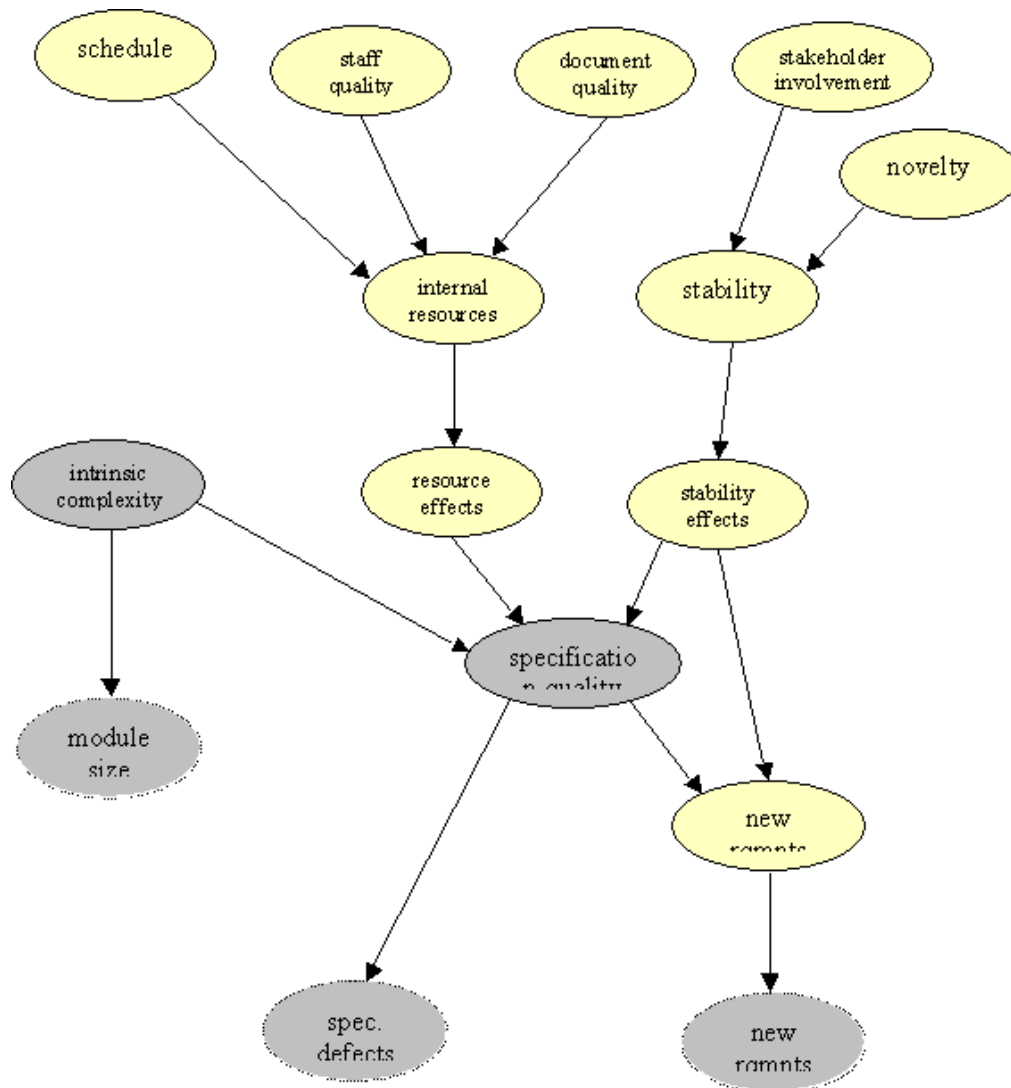


Figure FNK6: Specification quality sub-net.

With the above assumption the Specification quality sub-net can be explained in the following way: *specification quality* is influenced by three major factors:

- the *intrinsic complexity* of the module (this is the complexity of the requirements for the module, which ranges from “very simple” to “very complex”);
- the *internal resources* used, which is in turn defined in terms of the *staff quality* (ranging from “poor” to “outstanding”), the *document quality* (meaning the quality of the initial requirements specification document, ranging from “very poor” to “very good”), and the *schedule* constraints (ranging from “very tight” to “very flexible”);
- the *stability* of the requirements, which in turn is defined in terms of the *novelty* of the module requirements (ranging from “very high” to “very low”) and the *stakeholder involvement* (ranging from “very low” to “very high”). The stability

node is defined in such a way that low novelty makes stakeholder involvement irrelevant (Philips would have already built a similar relevant module), but otherwise stakeholder involvement is crucial.

The *specification quality* directly influences the number of *specification defects* (which is an output node with an ordinal scale that ranges from 0 to 10 – here “0” represents no defects, whilst “10” represents a complete rewrite of the document). Also, together with *stability*, specification quality influences the number of *new requirements* (also an output node with an ordinal scale ranging from 0 to 10) that will be introduced during the development and testing process. The other node in this sub-net is the output node *module size*, measured in Lines of Code (LOC). The position taken when constructing the model is that module size is conditionally dependent on *intrinsic complexity* (hence the link). However, although it is an indicator of such complexity the relationship is fairly weak - the Node Probability Table (NPT) for this node models a shallow distribution.

The Requirements match sub-net

The Requirements match sub-net (Figure FNK7) contains just three nodes. These could have been incorporated into the specification quality sub-net, but we have separated them out as a sub-net to highlight the overall importance that we attach to the notion of *requirements match*. This crucial output variable (ranging from poor to very good) represents the extent to which the implementation matches the real requirements. It is influenced by the number of *new requirements* and the quality of *configuration and traceability management*. When there are new requirements introduced and if the quality of configuration and traceability management is poor, then it is likely that the requirements match will be poor. This will have a negative impact on all subsequent testing phases (hence this node is input to three other sub-nets that model testing phases). For example, if the requirements match is poor then no matter how good the internal development is, when it comes to the integration and independent testing phases the testers will inevitably be testing the wrong requirements.

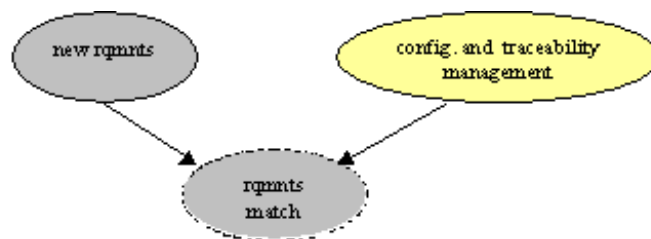


Figure FNK7: Requirements match sub-net.

The Specification Review and Test Process sub-nets

The Specification Review, Unit, Integration and Independent testing process, and Operational usage sub-nets are all based on a common testing idiom (Figure FNK8). The

basic structure of each is that they receive defects from the previous life-cycle phase as 'inputs', and the accuracy of testing and rework is dependent on the resources available. The 'output' in each case is the unknown number of residual defects, which is simply the number of inserted defects minus the number of discovered defects.

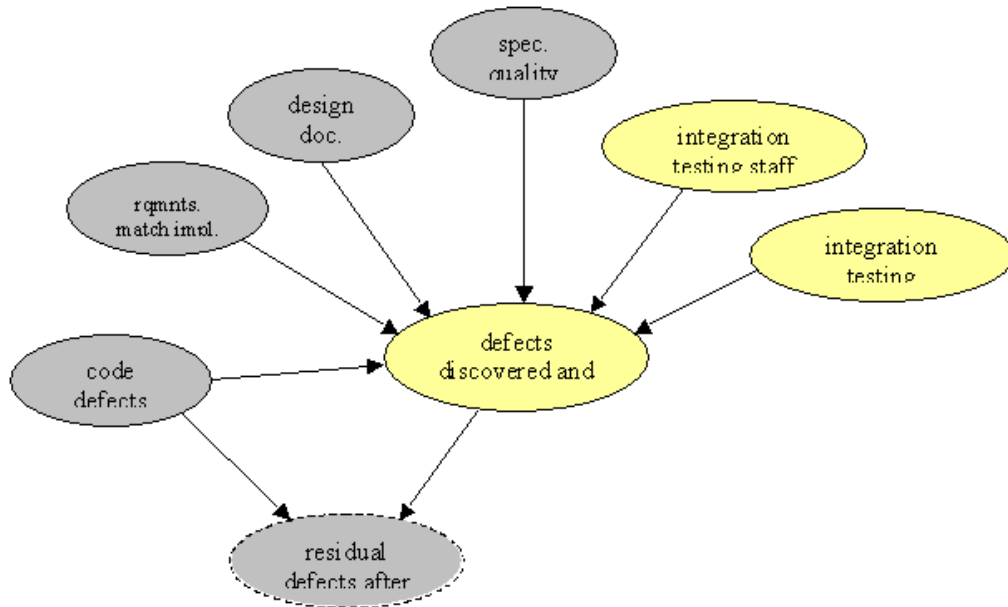


Figure FNK8: Integration testing process sub-net. This is an example of the generic testing idiom.

Design and coding process sub-net

The Design and coding process sub-net (Figure FNK9) is an example of the so-called “process-product” idiom. Based on various input resources something is being produced (namely design and code) that has certain attributes (which are the outputs of the sub-net). The inputs here are *specification quality* (from the specification quality sub-net), *development staff quality* and *resources*. These three variables define the *design and coding quality*. The output attributes of the design and coding process are the *design document quality* and the crucial number of *code defects introduced*. The latter is influenced not just by the quality of the design and coding process but also by the number of residual specification defects (an input from the specification sub-net).

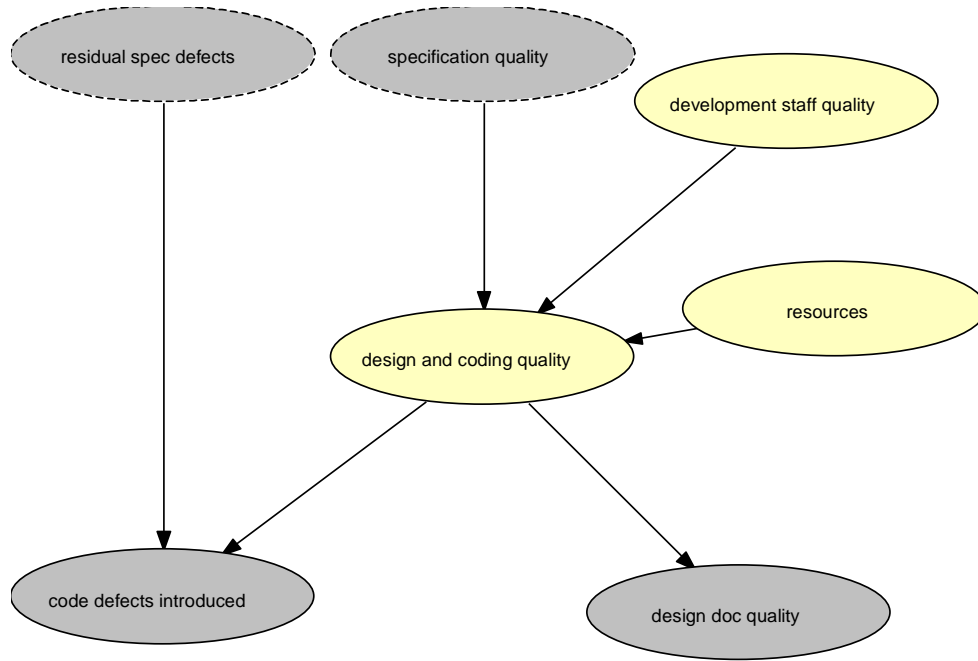


Figure FNK9: Design and coding process sub-net – an example of the “process-product” idiom

Defect density sub-net

The final sub-net is the Defect density sub-net (Figure FNK10). This sub-net simply computes the industry standard defect density metric in terms of residual defects delivered divided by module size. Notice that *defect density* is an example of a node that is related to its parents by a deterministic, as opposed to a probabilistic, relationship. This ability to incorporate deterministic nodes was an important contribution of the SERENE project.

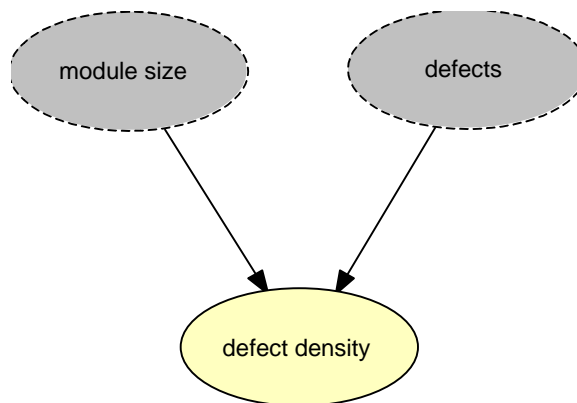


Figure FNK10: The Defect density sub-net.

The probability tables

The work on graphical probabilistic models described means that the problem of building such models now factorises into two stages:

- *Qualitative stage*: consider the general relationships between the variables of interest in terms of relevance of one variable to another in specified circumstances;
- *Quantitative stage*: numerical specification of the parameters of the model.

The numerical specification of the parameters means building Node Probability Tables (NPTs) for each of the nodes in the network. However, although the problem of eliciting tables on a node-by-node basis is cognitively easier than eliciting global distributions the sheer number of parameters to be elicited remains a very serious handicap to the successful building of probabilistic models. We will outline some of the techniques we used to handle this problem in this sub-section.

Note that for reasons of commercial sensitivity, the parameter values used in this paper may not correspond to the actual values used.

The leaf nodes (those with no parents) are the easiest to deal with since we can elicit the associated marginal probabilities from the expert simply by asking about frequencies of the individual states. For example, consider the leaf node *novelty* in Figure FNK6. This node has five states “very high”, “high”, “average”, “low”, “very low”. Suppose the expert judgement is that modules typically are not very novel, giving the following weights (as surrogates for the probability distribution), respectively, on the basis of knowledge of all previous modules in a development organisation:

5,10,20,40,20

These are turned into probabilities 0.05, 0.11, 0.21, 0.42, 0.21 (note the slight change of scale to normalise the distribution).

The NPTs for all other leaf nodes were determined in a similar manner (by either eliciting weightings or a drawing of the shape of the marginal distribution).

The NPTs for nodes with parents are much more difficult to define because, for each possible value that the node can take, we have to provide the conditional probability for that value with respect to every possible combination of values for the parent nodes. In general this cannot be done by eliciting each individual probability – there are just too many of them (there are several million in total in this BBN). Hence we used a variety of methods and tools that we have developed in recent projects [22, 23]. For example, consider the node *specification quality* in Figure FNK6. This has three parent nodes *resources*, *intrinsic complexity*, and *stability* each of which takes on several values (the former two have 5 values and the latter has 4). Thus for each value for specification quality we have to define 100 probabilities. Instead of eliciting these all directly we elicit a sample, including those at the ‘extreme’ values as well as typical, and ask the expert to provide the rough shape of the distribution for specification quality in each case. We then generate an actual probability distribution in each case and extrapolate distributions for all the intermediate values. To see how this was done, Table FNK1 shows the actual data we elicited in this case. The first three columns represent the specific sample values and

the final column is the rough shape for the distribution of “specification quality” given those values.

In the “best case” scenario of row 2 (resources good, stability high, complexity low) the distribution peaks sharply close to 5 (i.e. close to “best” quality specification). If the complexity is high (row 3) then the distribution is still skewed toward the best end, but is not as sharply peaked. In the “worst case” scenario of row 3 (resources bad, stability low, complexity high) the distribution peaks sharply close to 1 (i.e. close to “worst” quality specification).

On the basis of the distributions drawn by the expert we derive a function to compute the mean of the specification quality distribution in terms of the parents variables. For example, in this case the mean used was:

$$\text{min (resource_effects, (5 * resource_effects + intrinsic_complexity + 5 * stability_effects) / 11)}$$

In this example, to arrive at the distribution shapes drawn by the expert, we make use of intermediate nodes as described before. For example, there is an intermediate node *stability* which is the parent of the node *stability effects*. The *stability effects* node NPT is defined as the following beta distribution that is generated using the IMPRESS tool:

$$\text{Beta (2.25 * stability - 1.25, -2.25 * stability + 12.25, 1, 5)}$$

Figure FNK10 shows the actual distribution in the final BBN (using the Hugin tool) for the node specification quality under a number of the scenarios of Table FNK1. This figure provides a good consistency check – there is an excellent match of the distributions with those specified by the expert.

Resources (1 to 5 where 1 is worst 5 is best)	Stability (1 to 3 where 1 is worst 3 is best)	Intrinsic complexity (1 to 5 where 1 is most complex, 5 least)	Specification quality				
			1	2	3	4	5
5	3	1					
5	3	5					
1	1	1					
1	2	3					
1	3	5					
5	1	1					
1	1	5					

Table FNK1 Eliciting the probability table for *specification quality*.

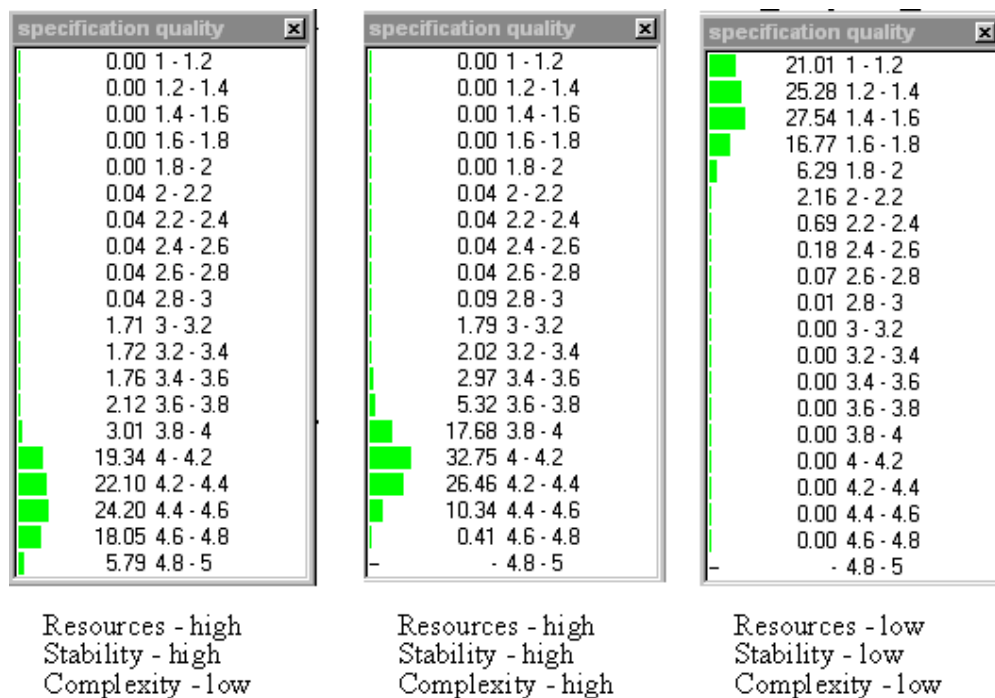


Figure FNK10: Actual distribution of specification quality for different scenarios

Some comments on the basic probabilistic network

The methods used to construct the reliability model have been illustrated in this section. The resulting network models the entire development and testing life-cycle of a typical software module. We believe it contains all the critical causal factors at an appropriate level of granularity, at least within the context of software development within Philips.

The node probability tables (NPTs) were built by eliciting probability distributions based on experience from within Philips. Some of these were based on historical records, others on subjective judgements. For most of the non-leaf nodes of the network the NPTs were too large to elicit all of the relevant probability distributions using expert judgement. Hence we used the novel techniques that have been developed recently on the SERENE and IMPRESS projects, to extrapolate all the distributions based on a small number of samples. By applying numerous consistency checks we believe that the resulting NPTs are a fair representation of experience within Philips.

As it stands, the network can be used to provide a range of predictions and “what-if” analyses at any stage during software development and testing. It can be used both for quality control and process improvement. However, two further areas of work were needed before the tool could be considered ready for extended trials. Firstly and most importantly, the network needed to be validated using real-world data. Secondly a more user-friendly interface needed to be engineered so that (a) the tool did not require users to have experience with probabilistic modelling techniques, and (b) a wider range of reporting functions could be provided. The validation exercise will be described in the next section in a way that illustrates how the probabilistic network was packaged to form the AID tool (AID for “Assess, Improve, Decide”).

Validation of the AID Tool

Method

The Philips Software Centre (PSC), Bangalore, India, made validation data available. We gratefully acknowledge their support in this way. PSC is a centre of excellence for software development within Philips, and so data was available from a wide range of projects from the various Business Divisions within PSC.

Data was collected from 28 projects from three Business Divisions:

- Mainstream Consumer Electronics,
- Philips Medical Systems
- Digital Networks.

This gave a spread of different sizes and types of projects. Data was collected from three sources:

- Pre-release and post-release defect data was collected from the “Performance Indicators” database.
- More extensive project data was available from the Project Database.
- Completed questionnaires on selected projects.

In addition, the network was demonstrated in detail on a one to one basis to three experienced quality/test engineers to obtain their reaction to its behaviour under a number of hypothetical scenarios.

The data from each project was entered into the probabilistic model. For each project:

1. The data available for all nodes prior to the Unit Test sub-net was entered first.
2. Available data for the Unit Test sub-net was then entered, with the exception of data for defects discovered and fixed.
3. If pre-release defect data was available, the predicted probability distribution for defects detected and fixed in the unit test phase was compared with the actual number of pre-release defects. No distinction was made between major and minor defects – total numbers were used throughout. The actual value for pre-release defects was then entered.
4. All further data for the test phases was then entered where available, with the exception of the number of defects found and fixed during independent testing (“post-release defects”). The predicted probability distribution for defects found and fixed in independent testing was compared with the actual value.
5. If available, the actual value for the number of defects found and fixed during independent testing was then entered. The prediction for the number of residual defects was then noted.

Unfortunately, data was not available to validate the operational usage sub-net. This will need data on field call-rates that is not currently available.

Given the size of the probabilistic network, this was insufficient data to perform rigorous statistical tests of validity. However, it was sufficient data to be able to confirm whether or not the network’s predictions were reliable enough to warrant recommending that a more extensive controlled trial be set up.

Summary of results of the validation exercise

Overall there was a high degree of consistency between the behaviour of the network and the data that was collected. However, a significant amount of data is needed in order to make reasonably precise predictions for a specific project. Extensive data (filled questionnaire, plus project data, plus defect data) was available for seven of the 28 projects. These seven projects showed a similar degree of consistency to the project that will be studied in the next sub-section. The remaining 21 projects show similar effects, but as the probability distributions are broader (and hence less precise) given the significant amounts of “missing” information, the results are supportive but less convincing than the seven studied in detail.

It must be emphasised that all defect data refers to the total of major and minor

defects. Hence, residual defects may not result in a “failure” that is perceptible to a user. This is particularly the case for user-interface projects.

Note also that the detailed contents of the questionnaires are held in confidence. Hence we cannot publish an example of data entry for the early phases in the software life cycle. Defect data will be reported here, but we must keep the details of the project anonymous.

An example run of AID

We will use screen shots of the AID Tool to illustrate both the questionnaire based user interface, and a typical validation run.

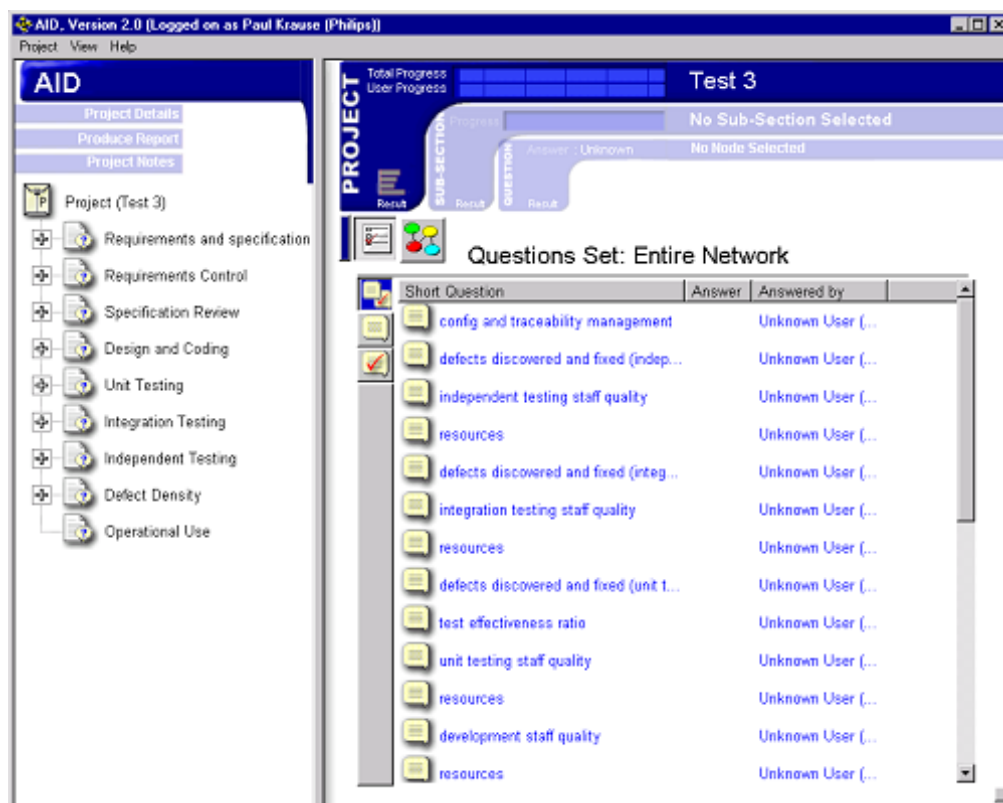


Figure FNK11: The entire AID network illustrated using a Windows Explorer style view.

One of the concerns with the original network is that many of the nodes have values on a simple ordinal scale, range from “very good” to “very poor”. This leaves open the possibility that different users will apply different calibrations to these scales. Hence the reliability of the predictions may vary, dependent on the specific user of the system. We address this by providing a questionnaire based front-end for the system. The ordinal values are then associated with specific question answers. The answers themselves are phrased as categorical, non-judgemental statements.

The screen in Figure FNK11 shows the entire network. The network is modularised

so that a Windows Explorer style view can be used to navigate quickly around the network. Check-boxes are provided to indicate which questions have already been answered for a specific project.

The questions associated with a specific sub-net can then be displayed. A question is answered by selecting the alternative from the suggested answers that best matches the state of current project. Figure FNK12 shows the question and alternative answers for the Configuration and Traceability Management node in the Requirements Control sub-network.

network.

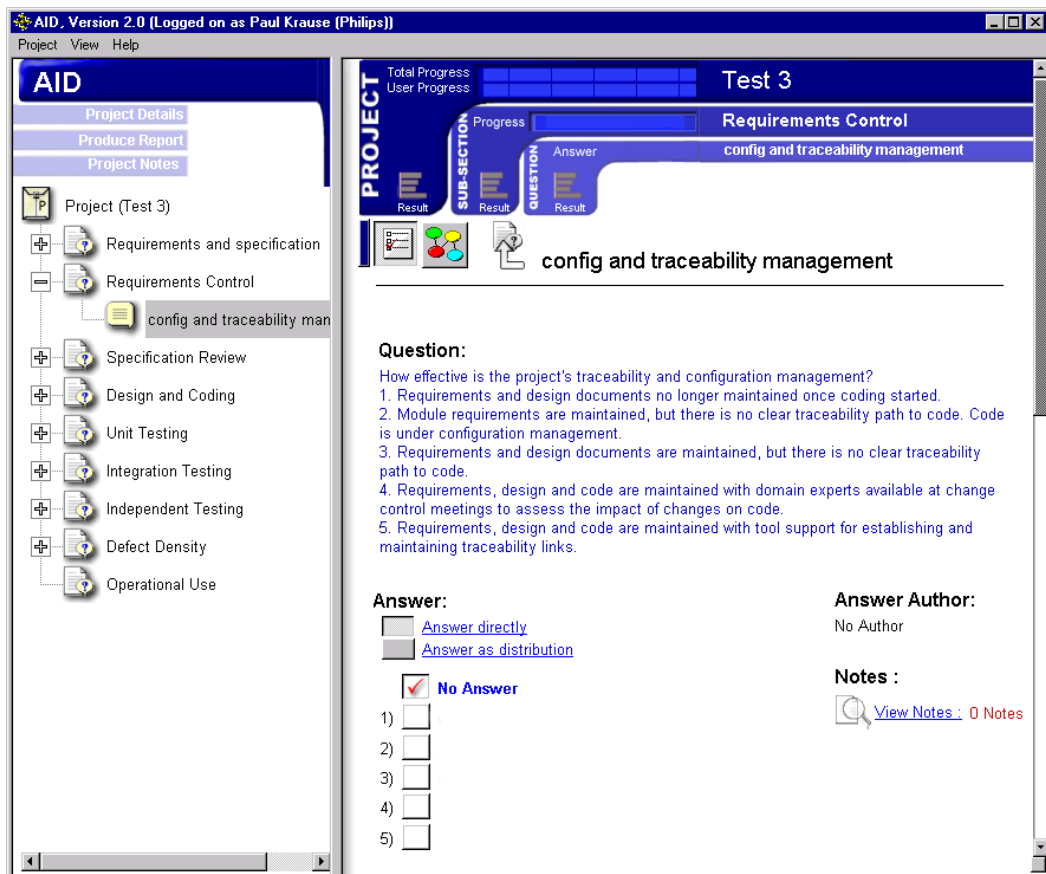


Figure FNK12: The question associated with the Configuration and Traceability Management node.

For this example project, answers were available for 13 of the 16 questions preceding “defects discovered and fixed during unit test”. Once the answers to these questions were entered, the predicted probability distribution for defects discovered and fixed during unit test had a mean of 149 and median of 125 (see Figure FNK13 – in this figure the monitor window has been displayed in order to show the complete probability distribution for this prediction. Summary statistics can also be displayed.). The actual value was 122. Given that the probability distribution is skewed, the median is the most appropriate summary statistic, so we actually see an apparently very close agreement between predicted and actual values. This agreement was very surprising as although we were optimistic that the “qualitative behaviour” of the network to be transferable from

organisation to organisation, we were expecting the scaling of the defect numbers to vary. Note, however, that the median is an imprecise estimate of the number of defects – it is the centre value of its associated bin on the histogram. So it might be more appropriate to quote a median of “100-150” in order to make the imprecision of the estimate explicit.

The actual value for defects discovered and fixed was entered. Answers for “staff quality” and “resources” were available for the Integration Test and Independent Test sub-networks. Once these had been entered, the prediction for defects discovered and fixed during independent test had a mean of 51, median of 30 and standard deviation of 45 (see Figure FNK14). The actual value was 31.

As was the case with unit test, there was close agreement between the median of the prediction and the actual value. “Test 3” was developed by PSC as a module or sub-system for a specific Philips development group. The latter then integrated “Test 3” into their product, and tested the complete product. This is the test phase we refer to as Independent Test.

The code size of Test 3 was 144 KLOC. The modules (perhaps sub-system is a better term given the size) used in the validation study ranged in size from 40-150 KLOC. The probabilistic reliability model incorporates a relatively weak coupling between module size and numbers of defects. The results of the validation continue to support the view that other product and process factors have a more significant impact on numbers of defects.

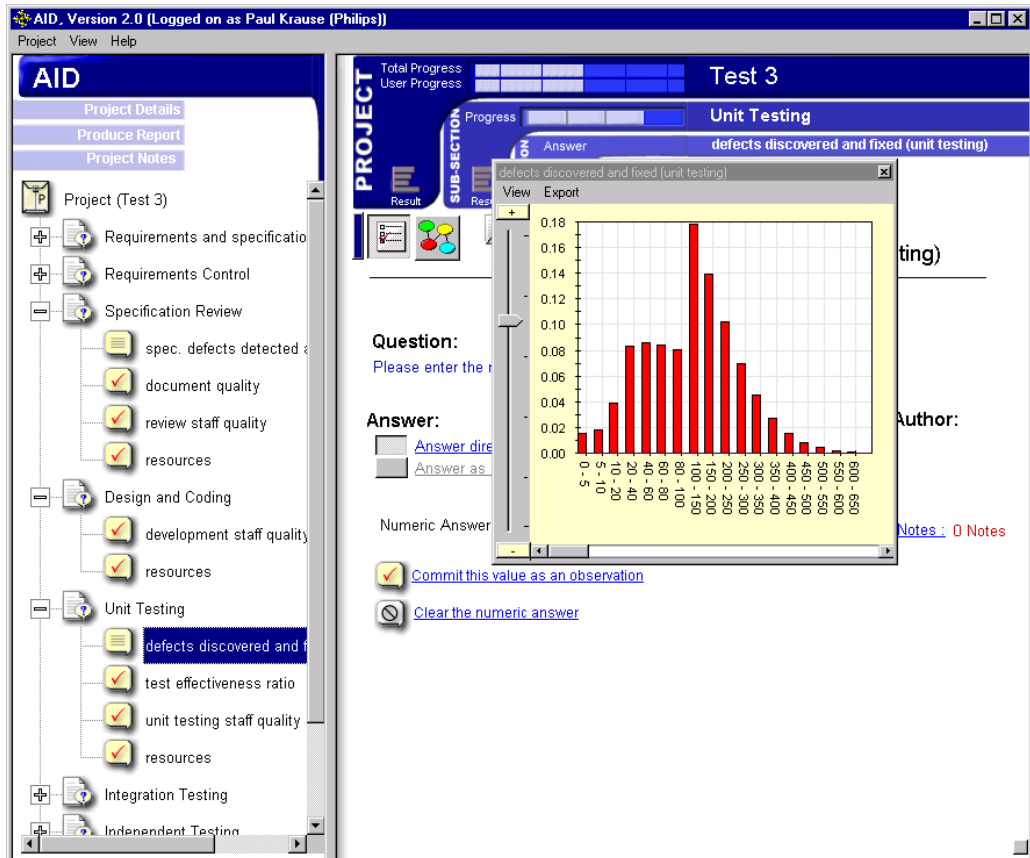


Figure FNK13: The prediction for defects discovered and fixed during Unit Test for project “Test 3”.

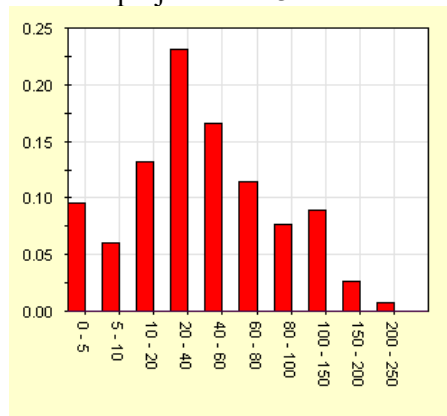


Figure FNK14: The prediction for defects discovered and fixed during Independent Test for project “Test 3”.

Conclusions

A disadvantage of a reliability model of this complexity is the amount of data that is needed to support a statistically significant validation study. This amount of data was not available as the metrics programme at PSC is in its early stages (as is the organisation

itself). Consequently, we were only able to carry out a less formal validation study. Nevertheless, the outcome of this study was very positive. Feedback was obtained on various aspects of the functionality provided by the AID interface to the reliability model, yet the results indicated that only minor changes were needed to the underlying model itself. We are now preparing for a more extended trial using a wider range of projects. This should begin in the early part of 2001.

Summary

We have described a probabilistic model for software defect prediction. This model can not only be used for assessing ongoing projects, but also for exploring the possible effects of a range of software process improvement activities. If costs can be associated with process improvements, and benefits assessed for the predicted improvement in software quality, then the model can be used to support sound decision making for Software Process Improvement (SPI).

The model performed very well in our preliminary validation experiments. In addition, a user interface has been developed for the tool that enables it to be easily used in a variety of different modes for product assessment and SPI. Although we anticipate that the model will need additional refinement as experience is gained during extended trials, we are confident that it will make a significant contribution to sound and effective decision making in software development organisations.

Acknowledgements

Simon Forey at Agena Ltd carried out the user interface development for AID; his contribution to this project was invaluable. We would also like to thank Mr. S. Nagarajan, Mr. Thangamani N., Mr. Soumitra Lahiri, Mr. Jitendra Shreemali and all the others at PSC, Bangalore who helped in the validation study. The validation study was started with Mainstream CE projects at PSC, with valuable support from the Software Quality Engineering Team in this division.

References

- [1] N. Fenton and M. Neil "A Critique of Software Defect Prediction Research", *IEEE Trans. Software Eng.*, 25, No.5, 1999.
- [2] N. Fenton, B. Littlewood, M. Neil, L. Strigini, A. Sutcliffe and D. Wright, "Assessing dependability of safety critical systems using diverse evidence", *IEE Proceedings on Software Engineering*, 145, No.1, February, 1998.
- [3] M. Neil, B. Littlewood and N. Fenton, "Applying Bayesian Belief Networks to Systems Dependability Assessment". *Proceedings of Safety Critical Systems Club Symposium*, Leeds, Published by Springer-Verlag, 6-8 February 1996.
- [4] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, (2nd Edition), PWS Publishing Company, 1997.
- [5] McCall, P.K. Richards and G.F. Walters, *Factors in software quality*.

- Volumes 1, 2 and 3. Springfield Va., NTIS, AD/A-049-014/015/055, 1977.
- [6] P. Ayton and E. Pascoe, "Bias in human judgement under uncertainty?" *The Knowledge Engineering Review*, 10, No.1, 21-42,1995.
- [7] W. Casscells, A. Shoenberger and T. Grayboys, "Interpretation by physicians of clinical laboratory results". *New England Journal of Medicine*, 299, 999-1000, 1978.
- [8] F. Akiyama, "An example of software system debugging", *Inf. Processing*, 71, pp. 353-379, 1971.
- [9] A.E. Ferdinand, "A theory of system complexity". *Int. J. General Syst.* 1, pp.19-33, 1974.
- [10] M.H. Halstead, *Elements of Software Science*. Elsevier North-Holland, 1975.
- [11] L.M. Ottenstein, "Quantitative estimates of debugging requirements", *IEEE Trans. Software Eng.*, 5:5, 504-514, 1979.
- [12] P. Hamer and G. Frewin, "Halstead's software science: a critical examination", *Proc. 6th Int. Conf. Software Eng.*, pp.197-206, 1982.
- [13] V.Y. Shen, S.D. Conte, and H. Dunsmore, "Software science revisited: a critical analysis of the theory and its empirical support", *IEEE Trans. Software Eng.*, 9:2, pp155-165, 1983.
- [14] M.J. Shepperd, "A critique of cyclomatic complexity as a software metric", *Softw. Eng. J.*, 3:2, pp 30-36, 1988.
- [15] T.M. Khoshgoftaar and J.C. Munson, "Predicting software development errors using complexity metrics". *IEEE J of Selected Areas in Communications*, 8:2, pp.253-261, 1990.
- [16] L. Hatton, "Re-examining the Fault Density-Component Size Connection", *IEEE Software*, March/April, 14:2, pp 89-98, 1997.
- [17] J Adams, *Risk*, UCL Press, 1995.
- [18] D. Heckerman. "A tutorial on learning Bayesian networks". *Technical Report MSR-TR-95-06*, Microsoft Research, March 1995.
- [19] P.J. Krause. "Learning Probabilistic Networks", *Knowledge Engineering Review*, 13, 321-351, 1998.
- [20] S.L. Lauritzen and D.J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)" *J. Roy. Stat. Soc. Ser B* 50, pp. 157-224, 1988.
- [21] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* Morgan Kaufman, 1988. (Revised in 1997)
- [22] SERENE consortium, "SERENE (SafEty and Risk Evaluation using bayesian Nets): Method Manual", ESPRIT Project 22187, <http://www.dcs.qmw.ac.uk/~norman/serene.htm>, 1999.
- [23] IMPRESS (IMproving the software PRocESS using bayesian nets) EPSRC Project GR/L06683, http://www.csr.city.ac.uk/csr_city/projects/impress.html.
- [24] Neil M, Fenton NE, Nielsen L, "Building large-scale Bayesian Networks", *The Knowledge Engineering Review*, 15(3), 257-284, 2000.

Authors and Company Information

Norman Fenton

Agena Ltd
11 Main Street
Caldecote
Cambridge, CB3 7NU
Tel: +44 (0) 1223 263880
Fax: +44 (0) 1223 263899
E-mail: norman@agena.co.uk

Norman Fenton is Professor of Computing at Queen Mary (University of London) and is also Managing Director of Agena, a company that specialises in risk management for critical systems. Between 1989 and March 2000 he was Professor of Computing Science at the Centre for Software Reliability, City University. Norman is a Chartered Engineer (member of the IEE) and a Chartered Mathematician (Fellow of the IMA). He has been project manager and principal researcher in many major collaborative projects in the areas of: software metrics; formal methods; empirical software engineering; software standards, and safety critical systems. His recent research projects, however, have focused on the use of Bayesian Belief nets (BBNs) and Multi-Criteria Decision Aid for risk assessment. Also, Agena has been building BBN-based decision support systems for a range of major clients. Norman's books and publications on software metrics and formal methods are widely known in the software engineering community. Norman is a member of the Editorial Board of both the Software Quality Journal and the Journal of Empirical Software Engineering, and is on the Advisory Board of Q-Labs.

Martin Neil

Agena Ltd
11 Main Street
Caldecote
Cambridge, CB3 7NU
Tel: +44 (0) 1223 263880
Fax: +44 (0) 1223 263899
E-mail: martin@agena.co.uk

Martin Neil is a Director of Agena Ltd. He holds a degree in 'Mathematics for Business Analysis' from Glasgow Caledonian University and a PhD in 'Statistical Analysis of Software Metrics' jointly from South Bank University and Strathclyde University. From 1996 - 1999 he spent four years at the Centre for Software Reliability, City University (London) and from 1992 - 1995 worked for Lloyd's Register as a consultant. His interests cover applications and theory in Bayesian probability, intelligent personalisation of media content, software reliability and risk management. Martin is a Chartered Engineer (member of IEE), the IEEE Computer Society and the

ACM. Through Agena Martin has consulted widely for many top organisations including Philips Research Labs., Motorola Research Labs., DERA (UK Defence Evaluation and Research Agency), Aon Risk Consultants, NATS (National Air Traffic Services, Civil Aviation Agency) and Railtrack plc

Paul Krause

Philips Research Laboratories,
Crossoak Lane, Redhill
Surrey RH1 5HA
Tel: +44 (0) 1293 815298
Fax: +44 (0) 1293 815500
E-mail: paul.krause@philips.com

Paul Krause has a varied background in software engineering. His areas of expertise include requirements management, formal specification, theoretical aspects of artificial intelligence, and software process and product quality assessment and improvement. He now works in the Specification and Testing Cluster of the Software Engineering and Applications Group at Philips Research Laboratories, Redhill, UK. Current projects address automatic test case execution for embedded systems, requirements management using the UML methodology, and software reliability prediction. Paul is also on the British Computer Society's ISEB Software Testing Board. From the beginning of 2001, Paul will be working part of his time as Research Professor in Software Engineering at the Department of Computer Science, Surrey University. Paul is a Chartered Mathematician (Fellow of the IMA).

Roger Shaw

Department of Computer Science
Queen Mary, University of London
Mile End Road
London
E1 4NS
E-mail: roger@dcs.qmw.ac.uk

Roger is a Visiting Research Fellow at the Department of Computer Science, Queen Mary (University of London) and a Senior Consultant with ERA Technology Ltd of Leatherhead, Surrey. Roger has worked in the computer industry for over 30 years. From 1969 he worked for the Health Service developing medical statistics applications. Roger joined ICL in 1973 and worked on the development of packet switching software (GANNET), protocol converters and real time and batch operating systems (George 3 and VME B). Roger then joined ITT (STC) working initially on PABX systems and then on the development and application of formal methods such as VDM and Z. In 1990 Roger joined Lloyd's Register where he managed their software research group, which addressed the application of formal methods and AI techniques to the development of safety critical systems. Roger joined ERA Technology in 1996 working initially on

safety critical systems and then as a department manager in charge of their very successful Y2K services. Latterly Roger has been placed in charge of ERA's software integration and testing services. Since being with Lloyd's Register Roger has been interested in decision support systems firstly through the DTI funded DATUM project and then through the ESPRIT funded SERENE project. Roger was instrumental in setting up the EPSRC funded SCORE project, which is looking at the use of Bayesian Belief Networks for modeling corporate, risk and safety cultures. This work is being conducted by the RADAR group at Queen Mary (London University) to which Roger is now affiliated.

Agena Ltd

Agena (www.agena.co.uk) specialises in providing decision support systems using Bayesian methods. Our decision support solutions include:

- TV Supreme — Personalisation and recommender solutions for Digital TV
- TRACS — Vehicle reliability prediction developed in collaboration with DERA. This system has broken new ground in its ability to predict vehicle reliability at the early design stages.
- AID — Consumer electronics fault prediction. Developed with Philips this system predicts software defects at various stages of development.
- BPAM — Assessing safety of electronic components. Developed with Railtrack this system helps choose between different contractors' offerings.
- RISKMAN Operational risk — our system can be tailored to predict different types of operational risk in different environments.
- Tailored decision support systems in the areas of Risk management and reliability assessment for business-critical and safety-critical computer-based systems.

Agena also offer consultancy in all aspects of Bayesian networks, software engineering and software metrics

Philips Electronics

Royal Philips Electronics is one of the world's biggest electronics companies and Europe's largest, with sales of EUR 37.9 billion in 2000. It is a global leader in color television sets, lighting, electric shavers, color picture tubes for televisions and monitors, and one-chip TV products.

Its 219,400 employees in more than 60 countries are active in more than 60 countries in the areas of lighting, consumer electronics, domestic appliances, components, semiconductors, and medical systems.

Session 9 - SPI and New Approaches

**Session Chair:
Howard Duncan, DCU, Ireland**

Patterns to Adopt Knowledge Based Solutions to Software Process Management Problems	9-2
Using Action Research as a Framework for an Improvement Program	9-15

Patterns to Adopt Knowledge Based Solutions to Software Process Management Problems

Risto Nevalainen, Director and Process Expert
Software Technology Transfer Finland STTF Ltd

Tel. +358-9-8811889, Fax +358-9-8045 1333

This article is based on article written by
Elixabete OSTOLAZA, Nuria QUINTANO and Giuseppe SATRIANI³
*European Software Institute ESI, Parque Tecnológico #204, E-48170 Zamudio,
Spain*

Tel. +34 944 209519; Fax +34 944 209420

Abstract. Organisations, both profit and non-profit, tend to “re-invent the wheel” when they come to management issues. In fact, previous experiences (even their own) seem very different (in form and context) from the current problems, so much so that the previous experiences do not appear to be valid in the problems solving process. In addition to the problem of validity of existing knowledge in new business situations, information on previous experiences is often available in different places, which increases the difficulties in sharing within the organisation or among different ones. The solution provided by PATTERNS will foster the culture inside organisations which supports readiness for understanding. This new culture approach is able to transform a company into “smart” organisation: knowledge based and learning. Specifically, PATTERNS allows the

introduction of the “learning from past experience” practice into the problems solving process of software companies.

1. Introduction

Problem solving consists many times of an approximation process to the final solution where the expertise and the ability to extrapolate conclusions from past experiences play a fundamental role. Both characteristics define what normally it's called an expert. An expert is a person who has special skill or knowledge in a particular field¹. According to Davenport and Prusak [1], knowledge is “a fluid mix of framed experience, values, contextual information, and expert knowledge providing a framework for evaluating and incorporating new experiences and information”.

Generally the expertise is tightly related to persons. What does it happen when this concept is translated to the organisations, that is to a group of persons organised for some end of work?¹ In organisations, “the knowledge is often embedded not only in documents, repositories, organisation routines, processes, norms and past experiences” [1] but it is also shared among the employee brains' and it constitute the intellectual capital for the organisation. Moreover, the knowledge is also highly dependent on the context of a particular experience or organisation, so much that the previous experience does not appear to be valid in the problem solving process.

Therefore the knowledge is very difficult firstly to recognise it, then to extract it and finally to package it in usable forms to accelerate the “learning from experience” process. This situation is worsened by the general reluctance of the organizations to share the knowledge gained in their daily work.

2. The Software Process Management context

Software Process Management (SPM) deals with the management of software development, maintenance and improvement related processes. SPM will never be an exact science because different variables can determine its success level. These variables mainly depend on the organisational context and on the organisational attitude to change. Another aspect that makes SPM more a guess than a controlled process is due to the absence of knowledge related to the experience made by software organisations in their attempt to improve their way of managing the processes affecting the software development and maintenance. This absence is even more absolute when someone looks for figures that can prove the benefit of SPM.

From the other side, there is available plenty of information, articles, methodologies, models, discussion forums, services, training, experts in software process management issues that instead of helping novices in getting them expert, create a dense jungle of chaos. In this context, the probability that a software organisation is knowledge-driven,

¹ Webster's Universal College Dictionary

learning and able to exploit the opportunities of an “*internetworked*” economy is quite low.

3. The solution provided by PATTERNS

PATTERNS aims to overcome the status-of-the-art by modeling the knowledge related to Software Process Management (SPM) to determine common characteristics and so derive cases that contain the contextualised knowledge, and the rules to be applied to store and retrieve them for being used in solving new problems.

The knowledge must be represented in a form that is really interpreted as useful information. As Liebowitz [2] says "good representation is essential if the knowledge is to be considered as valuable to the knowledge worker". For this reason, it is necessary to code the knowledge and express it in formal language to explicit it, and so many knowledge workers can share it. The PATTERNS project will use the patterns language [3] to packaging and transferring of knowledge useful for recognising a problem. ESI considers that from each available experience or from a sub-set of similar experiences could be derived a “case”. As Watson [4] has defined, a “case” is a contextualised piece of knowledge representing an experience, that is a “case” contains (see Figure 1) a solution for a problem in a context.

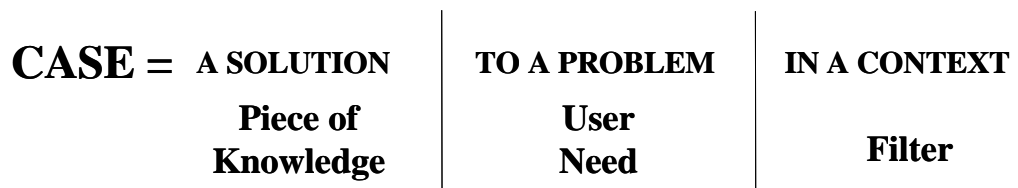


Figure 1 - Content of a CASE

PATTERNS aims to overcome the status-of-the-art by creating an *internetworked* architecture of Knowledge Centres² (KC) and distributed end-users, from which organisations can benefit so as to become knowledge driven and learning organisations. PATTERNS will provide a dynamically adaptive architecture that will allow organisations to increase their knowledge capacity, capitalise on that knowledge by transform it into business processes and manage the distributed knowledge embedded in software processes management practices.

PATTERNS aims to overcome the status-of-the-art by setting up a system that, through an intelligent query mechanism³, is able to provide good approximation to the solution of the SPM problem faced up by the user together with the rational of the provided solution. This process will include the interpretation of the user question, the

² A Knowledge Centre (KC) could be:

- an organisation with available experiences (explicitly collected or implicitly contained in experts’ brains)
- an organisation that facilitates the access of end-users to the PATTERNS network

³ A good example of such a kind of mechanism is the “Ask Jeeves” web-site: <http://www.ask.com/>

identification of the potential knowledge sources, the selection of the closest possible knowledge contained in each KC and in the external sources, the interaction with the user to capture new knowledge and then to learn from the experience.

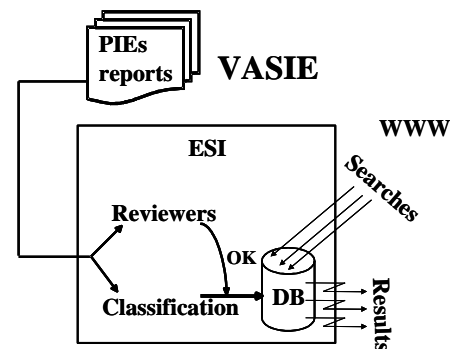
4. How PATTERNS was conceived

PATTERNS is a project funded by the Information Society Technology (IST) programme of the European Commission (EC). The project consortium currently includes the following European organisations: European Software Institute - ESI (Spain), INDRA Sistemas (Spain), YANA Research (Italy), TEKKVA Consult (Denmark) and Software Technology Transfer Finland - STTF (Finland). For detailed information about the project, visit the PATTERNS web-site: <http://www.esi.es/Patterns>.

The PATTERNS idea has gone evolving from 1996 together with the ESI expertise in relation to corporate repositories. The higher expertise ESI reached creating and managing these repositories the higher value ESI obtained analysing the information contained in these repositories. Major projects and services in ESI as predecessors of Patterns have been VASIE, ESI knowledge management infrastructure and SPI roadmap concept.

4.1 VASIE

When ESI acquired the leadership of the VASIE-2 project (ESPRIT/ESSI N. 24199) it was not totally aware of the value that could be obtained from it. This project is charge of disseminating, through a public database based on the WWW, the final reports of the Process Improvement Experiments (PIEs) funded by EC in ESSI programme (Ivth framework programme). A summarised explanation of the VASIE structure is described in Figure 2.



As long as the project was progressing and taking into account the type of VASIE users, the type of searches performed and the results obtained by another ESPRIT project (PERFECT⁴ – Project Number 9090), ESI perceived the big potentiality of the value contained in past experiences to solve current Software Process Improvement (SPI) problems. Under this project, ESI has been the catalyst of the European experiences of SPI, which has provided it the opportunity of participating in the ESPINODE⁵ initiative, and so reach even more expertise in this subject.

⁴ PIA Experience Factory – The PERFECT Handbook

⁵ ESPINODE - ESSI PIE nodes has been an European network of competent partners in order to stimulate, promote, and foster the use of Software Best Practices all over Europe

4.2 Repository of Expertise - The ESI knowledge management infrastructure

The possibility to provide an effective and accurate solution based on previous experiences depends on the quality and quantity of the available experiences: the larger is the experience base, the closer will be the solution for the user. To enlarge its knowledge base, ESI decided to define and create a Knowledge Management infrastructure [5], [6] to identify the sources of knowledge, collect and describe the identified knowledge, validate it, insert it in the knowledge base, transfer it and maintain its value.

4.3 Roadmaps for Software Process Improvement (SPI) and patterns of solutions

The analysis of the experiences available in VASIE made by ESI as contribution to the ESPINODE initiative has allowed to show commonality and variability among the actions performed in different experiences. Consequently, a certain number of patterns of solution for a specific software engineering problem or SPI roadmaps [7] have been identified. This achievement, together with the ESI expertise in domain modelling (usually applied to software reuse), has allowed to approach the knowledge management according to an innovative perspective: “the abstraction of patterns of solutions from available experiences”.

4.4 Knowledge management in other consortium companies

Similarly as ESI, also other consortium members have knowledge bases in their own focused areas. INDRA is a large software company having wide experience in software and system engineering areas and in project management. Their responsibility is to model their own internal knowledge and pilot Patterns solutions internally. Other partners are small, specialised companies doing SPI consultancy in their market segment or being experts in related technologies. STTF Oy has been long time in markets, providing software measurement and process assessment services in Scandinavia. Their expertise in SPICE and in project management is very relevant for Patterns.

5. PATTERNS Architecture

The solution proposed by PATTERNS allows users to obtain a rapid solution to a context-based problem. This solution will be provided according to the knowledge available locally at each Knowledge Centre (KC) and to the knowledge available in the network. PATTERNS will apply the general architectural solution (see Figure 3) to the specific context of organisations and end-users dealing with Software Processes Management practices. The choice of this domain has been motivated by the knowledge available in the project partners.

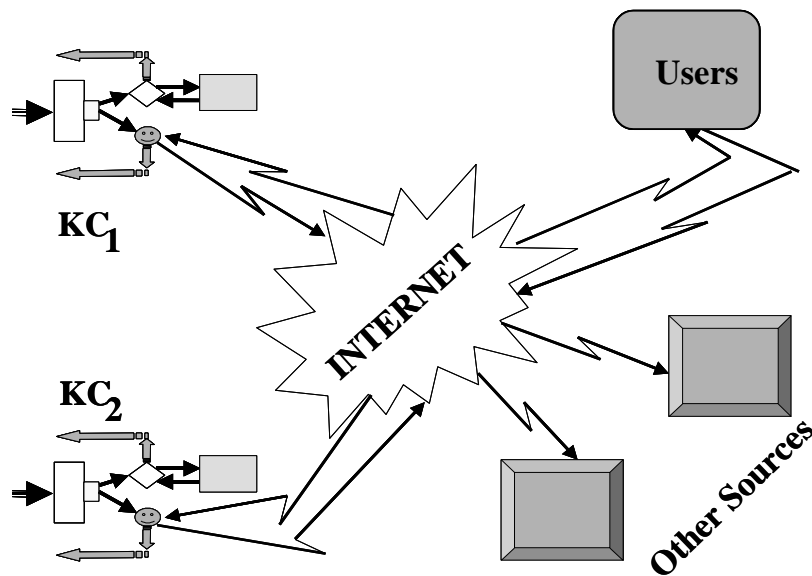


Figure 3 – PATTERNS Global Architecture:

- **Knowledge Centres (KC)**, where local knowledge repositories are active;
- **Internet**, that is the communication mean for remote search and connection of **Users**
- **Other Sources** of information and/or knowledge that could be useful for integrated extra elements for obtaining a more complete solution.

The main innovation of the PATTERNS solution is the integration of the main technologies described in the following sections of this article, into a user-friendly web-based application.

6. Knowledge Centres - Domain Modelling and Patterns Derivation

The Knowledge Centres (KC) involved in PATTERNS guarantee the provision of knowledge about Software Processes Management practices. But the knowledge existence is not sufficient to obtain knowledge useful, it is necessary to extract it and package it in usable forms. One of the first things that any science or engineering discipline must have is a vocabulary for expressing its concepts and a language for relating them together. A pre-requisite for defining a language is to identify the domains the language will model. It has been demonstrated in other application domains, that patterns can be a mechanism for domain modelling, for packaging and transferring of knowledge and for recognising a problem, based on the available solutions/experiences. As Richard Gabriel states [8]: “Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in

that context, and a certain software configuration which allows these forces to resolve themselves”. Moreover, Christopher Alexander [3] provides this definition: “Each pattern describes a problem that occurs over and over again in our environment and then describes the core of the solutions to that problem in such a way that you can use this solution a million times over without ever doing it the same way twice”. Each pattern is, then, a rule that expresses a relationship among a certain context, problem and solution. Using the pattern form, the description of each case tries to capture the essential insight that it embodies, so that others may learn from it, and make use of it in similar situations.

Each case derived will be stored in the database of cases. This database should be organized into a manageable structure that supports efficient search and retrieval methods. A balance has to be found between storing methods that preserve the richness of cases and their methods for accessing and retraining those cases that match against the problem purposed.

7. Natural Language Processing

Any application dealing with real world has to tackle the problem of handling informal communication. Tackling this problem corresponds, from a different point of view, to the effort of allowing a more natural way of communication between human and artificial resources. Generally, the real problem for Natural Language Processing (NLP) based application is domain modelling and instantiating the link between domain knowledge and linguistic knowledge. PATTERNS offers a really challenging line of research and development since, independently from the application of a NLP engine, it addresses both the problems of domain modelling and reasoning over the domain. In other words, it will try to define how to develop the knowledge base which represents formally a given domain. Our idea is to use this knowledge base as the (formal) representation of the context underlying ongoing (informal) communication.

The knowledge base will contain a set of frames representing different situations that can occur in a given domain. The user will be able to describe to the system the current domain and a situation to be managed. The system will retrieve a matching or a similar scenario in its database and return it to the user as a possible solution strategy. NLP provides to users the advantage of interacting with the system in natural language; it doesn't force users to use pre-defined list of choices for describing a problem, and the user will be able to “negotiate” exchanging information in order to reach an agreement about the topic of discussion.

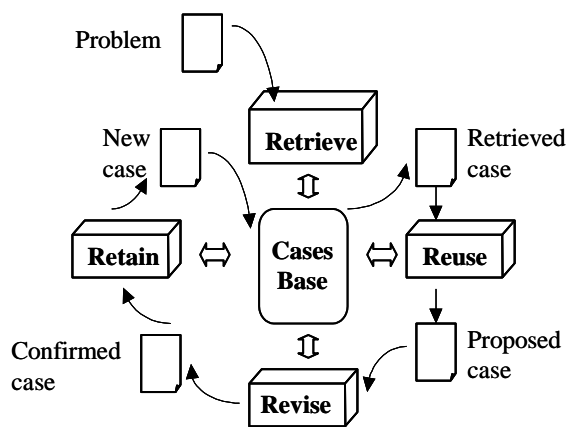
8. Case Based Reasoning

From a technological perspective, patterns are strongly related to Case Based Reasoning (CBR) technology, that helps to solve problems by storing, retrieving and adapting past situations or cases. In case-based systems a “case” is usually a contextualised piece of knowledge representing an experience [4]. It contains the past lesson that is the content of the case and the context in which the lesson can be used.

CBR makes direct use of past experiences in solving a new problem by recognising its similarity with a specific known problem and by applying its

solution to find a new solution for the current situation. An important characteristic of this technology is its capability for learning from the available knowledge, generating new cases. The procedure followed is complex but efficient; A new problem is matched against the available cases and one or more similar cases are retrieved. A solution suggested by the matching cases is then reused and tested for success. Unless the retrieved case is a close match, the solution will have to be revised, producing a new case that can be retained [4]. The capability of learning of the CBR tool is based on the use of heuristics, that allow modifying previous cases to fit new cases according to the user feedback collected after revising phase.

Figure 4 – Case Based Reasoning cycle



The mental process to describe CBR is typically represented by a cyclical process (see Figure 4) that receives as input the current problem description and:

- RETRIEVE the most similar case(s).
- REUSE the case(s) to attempt to solve the problem.
- REVISE the proposed solution if necessary.
- RETAIN the new solution as a part of a new case.

9. Intelligent Agents

Intelligent Agents (IA) can be seen as robots that can be trained to move autonomously inside an environment. They make decisions that simulate human behaviour in order to pursue the end-user's goals.

According to Liebowitz [2] an Intelligent Agent (IA) should be capable of adapting to user habits and preferences using learning techniques, also they

should decide when to help to user, what to help the user with, and how to help the user through its automated reasoning. And finally, an IA should can collaborate with other agents, exchanging information and service, and thereby solve problems that cannot be solved alone. The most important features of an IA are: intelligence and autonomy. Intelligence means that everything is oriented to user satisfaction by modelling human behaviour, learning from user history and providing an explanation of results. Autonomy means that user inputs are processed off-line and results are provided in real time if related with asynchronous events.

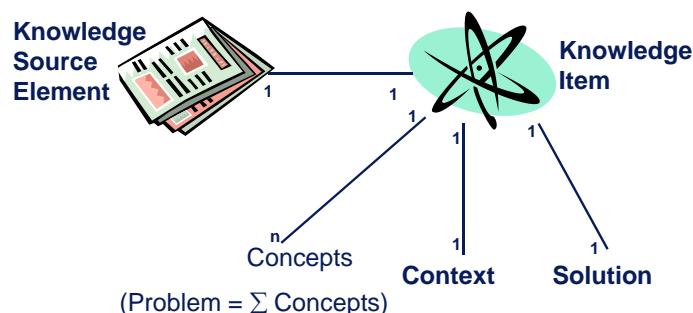
IA in the PATTERNS context are software units that scout, index and retrieve information to and from knowledge databases, and, at the same time, they are able to “learn” and enhance their search and retrieval capabilities and to provide new pieces of knowledge that, together with the interaction of the users, could allow the creation of new solutions.

10. Domain modelling

Domain modelling is a systematic way to identify, classify, extract and present knowledge in some defined domain. First selected domains in Patterns are SPI, project management and systems engineering. Also some kind of generic level of knowledge is more or less automatically included in Patterns solutions, because domains are overlapping and have many quite generic features. One example of that is classification scheme. One structured way to classify cases and user requests is SPICE capability scale (0...5, similar also in new CMMI model). It is quite obvious that such concepts will be used also in other domains, for example in safety and reliability engineering and in some industrial domains.

Result of domain modelling is packaged knowledge in knowledge items. These items are function of concept, context and solution, see Figure 5. Each of these elements is specified in detail in Patterns and will be tested in limited user community during 2002.

Figure 5 Main elements of domain modelling



Some knowledge source elements, like selected cases in VASIE database, are already classified and modeled. It is expected that many other sources will be available in future. Also some companies could use the same

concept and have their internal documents, quality system elements, metrics etc. well organised and more useful than before.

11. Some expected user scenarios of Patterns

Many potential user categories are identified in Patterns. Here some examples:

- **Patterns user** – Defines a problem to be solved and provides complementary information about the problem and its context, if needed. Gives feedback and refines retrieved solution.
- **Knowledge engineer** – Enters the initial information in the knowledge database. KE validates users' feedback and merges it with the main KI DB. Produces reports.
- **Knowledge transfer agent** – Uses Patterns as part of consultancy and training, to help customer in their transformation as intelligent company.

The purpose is that Patterns would be a commercial service after development and piloting phases during 2001 – 2001.

Some companies have already expressed their needs in SPI knowledge as open questions. It seems to be that there are two main dimensions: Some examples of different questions are collected from Finnish SPI community, see annex 1. The difficulty is to cover various needs and user requests as effectively as possible.

- Wide vs narrow questions. Some questions can near to business requirements and so typically wide. Some others can be very detailed and so narrow. See examples in annex 1.
- Open vs. structured questions. Some questions can be modelled easily and typical answers can be recorded and reused later. One typical such questions would be “What to do to improve from SPICE CL 2 to CL 3?” Even it can be difficult in some process and context, general answer is easy to give and so the question is well-structured. In opposite, some other questions are company specific or generic and so open by nature. See examples in annex 1.

12. Conclusions

The PATTERNS consortium thinks that the project will provide valid contributions to knowledge management and to Software Process Management (SPM).

The solution provided by PATTERNS will be applicable not only in SPM contexts but it will be very general and adaptable to different contexts. For this characteristic, it will foster the culture inside organisations which supports readiness for understanding. This new culture approach is able to transform a company into “smart” organisation: knowledge based and learning. This is a trend in the Information Society that leads to the introduction of strategies that bring together people, processes and technology, transforming the company culture into one that values learning and sharing.

Another new user requirement is gaining more and more importance within the Information Society; “How can I take advantage of the vast knowledge available, to find the information I need at the moment that I need it!”. PATTERNS proposes to decentralise the knowledge but to centralise the knowledge management. In this way users would have an homogeneous user interface to search across the available knowledge bases without having to know where the content is located.

SPM practitioners and SPM consultants aim to approach SPM in a more systematic way instead of as a pure guess. The definition of an SPM project and its probability to be successfully implemented in an organisation, should be based on historical data and on extrapolations. PATTERNS should become a repository of the SPM historical data, it should provide proved elements to start the extrapolation process and it should take advantage from the interaction with users to capture new knowledge to be integrated in the repository (it should learn from experience). Through an intelligent queries mechanism the system should provide good approximations to what users look for and possibly the rational for the solution provided.

Summarising, PATTERNS offers a new method of work that enables both individuals, but mainly, organisations to innovate and be more effective and efficient in their work. In this way, organisations will be able to increase their competitiveness by including past experiences in the current problem solving process.

References

- [1] T. DAVENPORT & L. PRUSAK, *WORKING KNOWLEDGE*. CAMBRIDGE, MA: HARVARD BUSINESS SCHOOL PRESS, 1998
-
- [2] J. Liebowitz: Knowledge Management Handbook CRC press LLC, 1999
- [3] C. Alexander: A Patterns Language: towns, buildings, constructions. Oxford University Press, 1977
- [4] I. Watson: Applying Case Base Reasoning: techniques for enterprise systems. Morgan Kaufmann Publishers, Inc., 1997
- [5] M. Blanco, J.A. Díez, P. Gutiérrez, L. Markaida, G. Satriani: Software Process Improvement Driven by Business Goals: The role of experiences. Sixth European Conference On Software Quality: Software Quality-The Way To Excellence. Vienna, Austria, April 12-16, 1999
-
- [6] M. Blanco, P. Gutiérrez, L. Markaida, G. Satriani: Repository of Expertise Guide. European Software Institute, ESI-1999-TR-017, 1999
- [7] M. Blanco, J.A. Díez, P. Gutiérrez, L. Marcaida, G. Satriani: Roadmaps for SPI. Submitted to IEEE Software magazine, accepted in February 2001
- [8] R. Gabriel: Patterns of Software: Tales From the Software Community. Oxford University Press, 1997

Annex 1. Some SPI questions and user requests in Finnish companies, august 2001

In table below I have collected some typical questions from various organisations. I have added some analysis like dimension wide – typical – narrow (classif 1) and open – semistructured – fully structured (classif 2). Also the typical domain in which question is closely related is presented.

Question	Classif 1	Classif 2	Classif 3
What practical actions are needed to move to levels 2 and 3 in SPICE?	Typical	Fully structured	SPI
How to motivate organisation to improve processes and find solutions to problems?	Wide	Open	SPI
What metrics we should use in SPI?	Typical	Semi-structured	Metrics
What criteria should be used to ensure project and risk management reviews?	Narrow	Semi-structured	Metrics
How to manage dependencies among projects and ongoing work packages?	Typical	Semi-structured	PM
Process for software package acquisition? What are the best practices? And risks?	Wide	Open	SPI
Can you provide a checklist to identify and assess project risks?	Narrow	Fully structured	PM
What is the right level of details in process modeling? What elements are enough in process model diagrams?	Typical	Semi-structured	SPI
Give metrics for process performance and effectiveness? How to validate SPI actions with metrics?	Wide	Semi-structured	SPI
How to present in process models the business needs ja policy?	Wide	Open	SPI
The best way for traceability in requirements - design - test cases?	Typical	Semi-structured	Metrics
How to automate calculations for remaining workload in a project?	Narrow	Fully structured	PM
Deployment of new processes and practices among staff and projects?	Wide	Open	SPI
How to define and collect process performance data? How to review and redefine metrics regularly?	Typical	Semi-structured	Metrics

Using Action Research as a Framework for an Improvement Program

Otto Vinter
Project Manager
Software Process Improvement
DELTA Software Engineering
DK-2970 Hørsholm, Denmark

Jan Pries-Heje
Associate Professor
Copenhagen Business School
DK-2000 Frederiksberg, Denmark

Abstract

At Brüel & Kjær, we developed a program for improving our requirements-engineering process that can serve as a guide to other companies as they set out to improve their own processes. Action research was our guiding strategy for formulating our improvement program framework. Action research embodies a strategy for studying change in organisations. This strategy consists of formulating a theory and an intervention strategy, and taking action to introduce change into the target organisation. We gained theoretical knowledge on effective requirements-engineering techniques, and then successfully put them to practical use in our organisation. We believe our program has applicability beyond improving requirements engineering.

1. Introduction

The difficulty of defining and executing improvement programs is well known and gives rise to many questions. Are we improving the right processes? How can we overcome resistance to the program? How can we ensure that improvements are diffused and

adopted throughout the organisation?

At Brüel & Kjær, we developed a framework for improving our requirements-engineering process that can serve as a guide to other companies as they set out to improve their own processes. Our framework is based on action research, which Bob Galliers [3] describes as an approach that lets researchers create new theoretical knowledge along with new methods that have practical value for the organisation.

Our framework consists of three phases. The phases evolved out of principles recommended by Susman and Evered [6]:

- Analysis phase,
- Focused pilot phase
- Broad dissemination phase

We will describe each of the three phases in detail. We first describe the analysis phase and how we found an optimum set of requirements-specification techniques. Next, we explain the focused pilot phase and the concepts underlying the main techniques that the projects used, followed by a discussion of the dissemination phase. Finally, we discuss how other companies might use our techniques to initiate their own improvement program.

We received funding for the analysis and the focused pilot phases from the European Union's European System and Software Initiative (ESSI) — our PRIDE project (A Methodology for Preventing Requirements Issues from Becoming Defects). The final report on this project is found in [9]. The broad dissemination phase was supported as part of a joint Danish effort of three universities, DELTA, and four companies – Centre for Software Process Improvement, and the results are documented in [4].

2. The Analysis Phase

The analysis phase consists of three activities:

- gather up-to-date information and diagnose problems,
- identify techniques for solving the problems, and
- prioritise the techniques using cost-benefit analysis.

Gather Information

Any analysis requires information to analyse. Such information might come from performing an assessment (e.g. CMM), from studies of literature, or, as in our case, from an analysis of problem reports from previous development projects.

Brüel & Kjær's software development process was widely seen as unsatisfactory. Too many projects had schedule overruns and products were often shipped with bugs. Even when management appointed task forces to improve product quality, problems were still reported from the field. The general opinion was that the main problem was insufficient testing before

release.

It had then been decided to perform an analysis of problem reports from previous projects to see, which types of problems were the most frequent. These analyses had shown that the testing process definitely needed improvement [8], but the largest cause for problems stemmed from the requirements engineering process.

Therefore, when we started the analysis of problem reports (what later became the analysis phase of our framework), the purpose was to find and implement effective techniques to improve the requirements engineering process. We performed an analysis of problem reports, with special emphasis on the problem reports related to requirements. We found that more than 50% of all problem reports could be classified as requirements related [9].

The primary issues seemed to be not in what was actually written in the requirements specification document, but rather in what was not known or tacitly assumed to be known about the requirements. This is demonstrated in Fig. OtV-JPH.1, in which we have grouped the requirements-related categories into: *Missing*, *Changed*, *Misunderstood*, and *Other*.

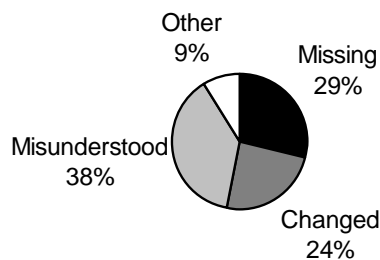


Fig. OtV-JPH.1: Major subcategories of requirements-related problems.

Our analysis further demonstrated that usability issues dominated in requirements-related problem reports (64%). This was something of a surprise. In the requirements engineering community, it is widely assumed that functionality issues are the major requirements problem, but our analysis showed that they represent only 22%.

We also found frequent problems (28%) in understanding and co-operating with third-party software packages and circumventing their bugs. This was the second largest requirements issue in the problem reports.

Identify Problem-Solving Techniques

While analysing and categorising the problem reports, we simultaneously carried out a

survey of the literature on requirements elicitation, specification, and use (cf. Davies [2], Sommerville & Sawyer [5], and Thayer & Dorfman [7]) and found several techniques recommended for achieving better requirements.

Next, in the process of problem-report analysis, we tried to imagine ways to prevent each bug. We began with our list of known techniques and added to it if none of the listed techniques could prevent the bug in question. We also considered and later dropped many well-known techniques because they seemed useless in relation to actual bugs. Many of the more useful techniques were “common sense” procedures that we moved up from the design phase to the requirements phase and then formalised. Used in this context, such techniques seem to contribute significantly to product quality.

Prioritise Techniques

Next, we performed a cost-benefit analysis on the techniques by estimating the hit rate of each technique for each error report. Because we had no data on the actual benefits of preventing a specific bug, we used the time-to-find-and-fix as the benefit. We estimated each technique’s savings by multiplying this benefit by the hit rate, accumulating over all error reports, and then subtracting the cost (time) of using the technique. Using this cost-benefit analysis, we produced a prioritised list of 13 techniques.

3. The Focused Pilot Phase

The focused pilot phase consists of three activities:

- let one or two pilot projects select techniques to implement,
- train the teams in the techniques, and
- follow up with teams and evaluate how they use techniques.

From a managerial viewpoint, it might seem faster to forego pilot projects and simply enforce the use of optimised techniques across the organisation. However, that kind of dissemination often fails. As Gerald Weinberg [10] puts it: “Attempts to change software organisations commonly fail because of inadequate understanding of change dynamics.” According to Weinberg, this model of instant change fails in reality because the model’s assumptions are wrong: “Forcing people to use techniques would probably create more resistance to change than actual change.”

Behaviour changes one person at a time, project by project. Because our project teams typically consist of three to five people, we limited our implementation efforts to two carefully selected projects. This choice proved extremely useful in disseminating the techniques throughout the organisation: Project members spread the word about improvements in an unplanned, but very effective way. Based on our experience, we recommend that you start with a focused pilot phase before you perform a broad dissemination.

Technique Selection

To ensure each pilot project's commitment to the techniques, we applied the theory of planned behaviour (cf. Ajzen [1]). According to this theory, "people's behaviour is strongly influenced by their confidence in their ability to perform." Furthermore, empirical analysis shows that when adopting new techniques, personal considerations tend to overshadow "the influence of perceived social pressure."

Given this, we first held a one-day introduction to our methodology and the top 13 techniques, then interviewed project participants individually to get their opinions on each technique. We then summarised the analysis and presented the results to the teams. Based on the results, each team selected five techniques, for a total of seven different techniques (see the box below, "Selected Requirements Engineering Techniques").

Selected Requirements Engineering Techniques

- *Scenarios*
Relate demands to use situations and describe the essential tasks in each scenario.
- *Usability Test of Functional Prototype*
Ensure that the system meets users' day-to-day needs by developing and user-testing a functional user-interface prototype.
- *Consult Product Expert*
Let a product expert check screens for deviations from earlier product styles.
- *External Software Stress Test*
Test to ensure that external software fulfils the requirement expectations, emphasising extreme cases.
- *Orthogonality Check*
Check requirement specification to ensure that users can apply operations and features whenever they are useful.
- *Initial Value Check*
Check to ensure that it is clear which attributes should appear when a screen is opened or an object is created.
- *Performance Specifications*
Ensure that the requirements specification contains performance goals for each requirement.

Train Teams to Use Techniques

We developed a two-day workshop to train project teams in the techniques they selected. Our focus was on preparing them to apply the techniques. We therefore structured the course so that project team members could try the techniques on problems related to their work. In one case, for example, a four-member project team developed a paper mock-up for their upcoming project and then usability tested it with a person from another group.

Follow Up and Evaluate Use

During the focused pilot phase, we interviewed a majority of the team members to assess their attitudes toward the techniques. The interviews took place at four major pilot-project milestones:

- immediately after training
- after scenarios were developed
- after usability tests (e.g. close to the end of the requirements phase)
- after the product release.

The questions in these interviews were always the same (see the box below, "Interview Guide"). The team members responded very positively to techniques they used on the pilot-projects.

Interview Guide

1. If you didn't use the technique, why was that?
2. What was your experience with using the technique?
3. Can you give us an example of how it was successfully used
4. Do you consider this technique useful and efficient?
5. Did the workshop give you enough knowledge to use the technique successfully?
6. How much extra time did it take to use this technique (compared with either not using it or with using a different technique in earlier projects)?

Both pilot projects completed the scenario technique as prescribed. However, once they'd written the scenarios, the teams could not wait for a functional prototype to be developed. Instead, in only two weeks they developed a prototype using a screen mock-up to show navigational facilities. One team did this in Visual Basic; the other used the Bookmark feature in MS Word 6.

Of the remaining three techniques, both teams failed to apply one technique and applied the other two techniques incompletely. Based on this, we realised that introducing so many new techniques at once was too ambitious. Two new techniques was all the teams achieved to perform properly.

At this point, one of the projects got a new project manager who didn't believe in the techniques. He discarded the prototypes and designed the user interface to resemble a product he was familiar with. Product development continued from there. We had little opportunity to study the project further; we know only that the team overshot their budget significantly and didn't complete the project on time.

Although the requirements-engineering process took longer than we expected, the specification and design phases were shorter than expected and thus there was no critical delay overall on the other project. Following the pilot-phase, the project continued product development according to Brüel & Kjør's standard software development procedures.

Once the product was released, we analysed the problem reports in the same way we had done before. We found a significant reduction in error reports and an impressive reduction in usability-related requirements issues for each new screen. However, the greatest impact of the requirements-engineering techniques was on users' perception of product quality. The product steadily sells more than twice as much as the team's earlier product.

4. The Broad Dissemination Phase

The broad dissemination phase consists of four activities:

- disseminate pilot-project results,
- diffuse knowledge about techniques,
- support projects in using techniques, and
- evaluate how projects use techniques

Disseminate Results

After our final evaluation of the pilot project, we presented our findings within Brüel & Kjør and also outside the company. We invited software developers from all major development projects to at least one presentation of the results. Furthermore, we posted all intermediate documents and presentation materials on the company intranet. Following this, we received many requests for copies of the training material from the pilot phase, especially material related to the scenarios.

Several projects then contracted with us to train them in the techniques. We trained four project teams in the second round of improving the requirements-specification process. Based on the convincing results from the pilot projects, the new project teams were motivated to change and knew which techniques they wanted to use. As a result, we could move right into the training stage.

Diffuse Knowledge about Techniques

Once again we held a two-day workshop for project team members. This time, however, we did not include the techniques that did not work well in the pilot phase; our course focused solely on scenarios, prototyping, and usability testing. Furthermore, we discouraged the teams from creating fully functional prototypes. That is, we gave them examples of simple, early development stage prototypes and emphasised how much they could achieve using them, offering examples from the pilot projects.

Support Technique Use

In the pilot phase, we interviewed the project members several times. During these interviews, they asked us questions that helped them better understand and apply the techniques. However, we didn't directly interact with the projects while they were using the techniques.

In the dissemination phase, we decided to keep close, continuous contact with the projects. We did this for two reasons. First, in the pilot phase, some techniques were not used correctly. Second, we hoped to avoid a repeat of the pilot-phase situation in which a project manager abandoned the techniques.

We arranged to meet regularly with each team during the analysis and requirements-specification phases. At these meetings, we listened to team members and offered to mentor them in using the techniques, thereby ensuring that they were applied in the best possible way. In doing this, we ensured that the projects stayed on track and did not cut corners in applying the techniques because of other problems in the project. Although we initially planned to meet monthly, we actually held meetings every two weeks or so.

At the meetings, we used a standard agenda, discussing:

1. project status and activities planned for the near future;
2. experiences, problems, and success with the techniques;
3. issues or areas that needed our support; and
4. when to hold the next meeting.

Each meeting typically took two hours and we also spent, on average, about four hours per team in between-meeting support.

Our direct interactions were a success. All four projects completed the requirements phase using the techniques. One of the projects was later cancelled for internal organisational reasons. Two other projects were significantly rescoped, but the scenario results and usability test experiences remained valid and were used in product development.

Evaluate Use of Techniques

Once the four projects had used the techniques, we again interviewed project participants. Next, we analysed the interview transcripts from all six projects. The following is a summary of what we found.

Our requirements engineering process improved significantly. The techniques we introduced are now standard practice in our company, and are supported with training material and documented in ISO 9001 procedures. Developers' reaction was very positive, and both product quality and sale figures increased.

Although we originally intended to try several requirements-engineering techniques, in the end our efforts were focused on scenarios and usability tests on early prototypes. These techniques proved to be very effective for several reasons. First, we could teach a team of developers to use the techniques in two days. Second, the techniques required neither expert knowledge nor support to achieve significant benefits. Finally, as we noted above, developers were extremely satisfied with the results they achieved.

In interviews, developers were enthusiastic about the opportunity to have closer contact

with potential customers and users. This contact gave them a wealth of domain knowledge, which in turn let them develop better and more user-friendly products. They also said that the techniques helped them more thoroughly perceive the real needs of potential customers and users.

Developers also repeatedly mentioned improvements in internal communication and co-ordination. In particular, they noted that there were far fewer exhausting discussions about how to interpret user requirements and needs. Time spent discussing different implementations also decreased because they could simply refer to a scenario or a specific validation of a prototype idea, which without dispute served as a common reference.

Because we conducted a pilot phase first, we were able to demonstrate the techniques' success to both developers and management. The pilot project's impressive results motivated other projects to use the techniques. It was never necessary to "sell" the techniques at Brüel & Kjær to get development teams to use them.

Finally, our direct and regular interaction with the project teams was an extremely effective tool for ensuring that the techniques were used—and used properly. This interaction ensured that the teams' experience in using the techniques would produce the expected results.

5. How Can Other Companies Improve?

Clearly, Brüel & Kjær gained much from this process improvement, and we believe other companies can benefit from our program. Although we do not have statistics to support this claim, we have presented our results to experienced developers outside our company and several have been inspired to start similar improvement efforts. Also, we believe our program has applicability beyond improving requirements engineering. For example, it could be used to improve project management, risk management, quality assurance, and so on. However, to prove this, further research and investigation is needed.

In eliciting the phases and their activities, we have tried to be as general as possible. The box below, "Overview of Framework for the Improvement Program," shows our framework's three phases and 11 activities. As the list shows, we added an activity—"support use of techniques in projects"—in the focused pilot phase. Although we did offer informal support for the pilot projects, we added systematic mentoring and regular support meetings in the broad dissemination phase. Today, we would definitely offer this formal support and mentoring in the pilot phase, too.

Overview of Framework for the Improvement Program

Analysis Phase

1. Gather up-to-date information and diagnose problems.
2. Identify techniques for solving the problems.
3. Prioritise the techniques using cost-benefit analysis.

Focused Pilot Phase

1. Let one or two pilot projects select techniques to implement.
2. Train the teams in the techniques.
3. *Support use of techniques in projects.*
4. Follow up with teams and evaluate how they use the techniques.

Broad Dissemination Phase

1. Disseminate pilot-project results.
2. Diffuse knowledge about the techniques.
3. Support projects in using the techniques.
4. Continuously evaluate how projects use the techniques.

Action research embodies a strategy for studying change in organisations. This strategy consists of formulating a theory and an intervention strategy, and taking action to introduce change into the target organisation. Action research was our guiding strategy for formulating our improvement program framework. We gained theoretical knowledge on effective requirements-engineering techniques, and then successfully put them to practical use in our organisation.

Could we have used other research approaches? Action research is just one of several qualitative research methods used in the field of information systems. World-wide, action research is eclipsed by more traditional social-science methods such as case studies, experiments, and sampling surveys [3]. However, none of the other research methods are well suited to introducing change in the course of a study. Thus, action research should be very important for the study of SPI because it is oriented toward change, especially in situations where participation and organisation-wide change is required. We believe that our study bears this out.

References

- [1] Ajzen, I. (1991). "The Theory of Planned Behavior." *Organizational Behavior and Human Decision Processes*. Vol. 50. pp. 179-211.
- [2] Davis, A.M. (1990). *Software Requirements: Analysis & Specification*. Upper Saddle River, N.J.: Prentice Hall.
- [3] Galliers, B. (1992). "Choosing Information Systems Research Approaches." Galliers, R., ed. *Information Systems Research: Issues, Methods and Practical Guidelines*. Oxford, U.K.: Blackwell Publishers.
- [4] Mathiassen, L. e.a. eds. (2001). *Improving Software Organizations: From Principles to Practice*, Addison-Wesley Crystal Series.
- [5] Sommerville, I. & P. Sawyer (1997). *Requirements Engineering: A Good Practice Guide*. New York: John Wiley & Sons.

- [6] Susman, G. & R. Evered (1978). "An Assessment of the Scientific Merits of Action Research." *Administrative Science Quarterly*. Vol. 23, No. 4. pp. 582-603.
- [7] Thayer, R.H. & M. Dorfman, eds. (1997). *Software Requirements Engineering*, second edition. Los Alamitos, Calif.: IEEE Computer Society Press.
- [8] Vinter O., P.-M. Poulsen, K. Nissen & J.M. Thomsen (1996): The Prevention of Errors through Experience-Driven Test Efforts. ESSI Project 10438. Final Report, Brüel & Kjær A/S, DK-2850 Nærum, Denmark. (<http://www.esi.es/ESSI/Reports/All/10438>).
- [9] Vinter O., S. Lauesen & J. Pries-Heje (1999): A Methodology for Preventing Requirements Issues from Becoming Defects. ESSI Project 21167. Final Report, Brüel & Kjær Sound & Vibration Measurement A/S, DK-2850 Nærum, Denmark. (<http://www.esi.es/ESSI/Reports/All/21167>)
- [10] Weinberg, G.M. (1997). *Quality Software Management/Volume 4: Anticipating Change*. New York: Dorset House.

About the Authors

Otto Vinter is a project manager with DELTA specialising in software process improvements. He has managed development projects for 30 years. He holds a M.Sc. in Computer Science from the Danish Technical University (1968). He presents, gives seminars, and consults. He also acts as an expert evaluator for EU research and development programmes (IST). The work reported here was performed while he was managing process improvement actions at Brüel & Kjær.

Jan Pries-Heje is a researcher and consultant. He has for many years been interested in leadership and organisational issues in relation to software development. He holds a M.Sc. in Computer Science and Business Administration (1989), and a Ph.D. in software development (1993). He is a certified ISO 9000 auditor and BOOTSTRAP assessor. Currently he is a Visting Professor at Georgia State University in Atlanta.

Session 10 - SPI and Supplier Management

**Session Chair:
Robert Olesen, Delta, Denmark**

Customer Relationship Management - A Maturity Model	10-2
Experiences On Outsourcing Requirements Specifications	10-16

Customer Relationship Management - A Maturity Model

Henrik Ekstam
KTH

Daniel Karlsson
KTH

Terttu Orci
Stockholm University/KTH

Introduction

Customers are the essentials of every business. Without customer satisfaction, a business runs out of its customers as soon as there are competitors on the market. Customer satisfaction is highly dependent of the product delivered, but also on how well the supplier maintains his relationship with the customer. The efforts invested in customer satisfaction should ideally return the maximum of the investment. Efforts should be made both to strengthen and broaden the relationship. Some customers are, however, more important than others in terms of potential return on investment. This suggests that not every customer should have the same treatment, but there should be consciousness of the strategy how different customers should be handled to make the most of the available resources for customer relationship management (CRM). CRM is the area for these issues, an area integrating people, technology and processes.

It is, however, not obvious how CRM work should be performed in an organisation. The set of interesting questions in a more global perspective are "What processes should be in place?", "Why?", "How to make most of the CRM activities, optimising the effort towards customer satisfaction and profit?", and "How to control the activities towards the desired goals?". These issues have clearly much in common with the well-known

software engineering problems; i.e. how to achieve predictability in time, cost and quality.

In software development, publicly available models exist for the purpose, SW-CMM (Paulk et al, 1995) being the most well known. It presents a roadmap to follow, towards predictability in terms of time, costs, and quality of the software projects. The roadmap is intended for a long-term use; to climb on the ladder systematically, introducing control and best practices on project level to start with, and later on the organisational level in a broad perspective. In the course of improvement activities, the organisational consciousness and knowledge is increased, the participatory culture will become a way of life, and the employees involved in development and management know what is expected from them in terms of task performance and responsibility.

Similar approach should be favourable also in CRM, with only small steps at a time, evolutionary rather than revolutionary, on the way towards a goal of high maturity.

As far as we know, there are no publicly available maturity models for CRM. In this paper, such a model is briefly presented. The model is presented in detail in (Ekstam, Karlsson 2001). The architecture of the model is based on the idea of CMM, the ladder of levels, with the underlying assumption that not the whole CRM should be introduced or changed at once, but stepwise. The intentions of CMM of increasing homogeneity and improvement have been used as a guideline for the model construction. The model is so far only a proposition, without any empirical validation. It has been constructed from empirical observations, literature studies and discussions with experts in the domain. The model has been presented to domain experts, and the spontaneous comments have been very positive. The obvious next step to take is to validate it in an industrial setting, a work already started with.

Customer Relationship Management

The basic thought behind CRM is quite simple: The most profitable customer is one who is loyal, buys a in large quantities and needs little or none marketing efforts. Provided with a sufficient number of such customers one wouldn't need to take the costs and efforts of sales and marketing. Unfortunately, very few, if any, business has these kinds of customers. But by paying attention to questions about loyalty and relations, your customers can be much more ideal. The process of transforming the customers is known as CRM.

There are several definitions of CRM. CapGemini Ernst & Young (CGEY, 2000) presents the following definition:

“CRM is a strategic and holistic approach to grow enduring relationships with profitable customers, in order to: increase the number of customers, achieve high retention rates, retain the most profitable customers, achieve a greater share of customers' spend, take a pro-active "customer view" rather than a "product view", build customer loyalty through intimate relationships and establish lifetime relationships with customers. In short, CRM means: Focus on the customer.”

Having a strategy is essential while working with CRM. This strategy work can be conducted, according to (Karlöf, 1996) in either of two ways:

- strategy without changing the environmental options
- business development in terms of new business opportunities

To improve/develop a strategy further is not always necessary. In stable business situations and conditions, the management might not have any need to develop the strategy and think in long-term manner. Naturally, when a problem occurs, it is solved and may lead to strategy improvement. However, a long-term strategy work, if initiated, should be conducted as a larger investment, as a process improvement work similar to software process improvement, in a number of steps:

- initiating
- data collection and analysis
- synthesis and formulation
- archival of strategy
- measurement and goal fulfilment
- continuous strategic management

In other words, the objectives of CRM are to create long-term, mutually valuable relations with important customers. The profit from these relations should be seen in a long-term perspective. There are, according to (Buttle, 2000) the following reasons for this:

- The competition of the customers is increasing, the advantage of local presence is decreasing as geographic borders are diminishing because of the business alliances. The pressure on logistics and distribution increases as the local presence decreases.
- The market is divided in segments more and more. The focus is moved from classical mass market with higher needs than available, towards more individualistic marketing, so called one-to-one marketing. That is based on the assumption that customers become more loyal if the offers are tailored to their specific needs, maybe unique needs.
- The customers are becoming more demanding. The expectations on reliable products and good service are increasing. The tolerance for faults is decreasing; comparisons are done with best practice. The expectations are shifting, what was OK yesterday is not satisfactory today.
- The product quality has generally increased during the last years and it is not self evident that the quality of the product can give competitive edge.

These reasons, combined with the increased possibilities with tailoring special solutions and offering them to customers is a good starting point. The internet, software and phones makes it easy to cultivate closer relationships with the customers.

There is lifetime value (LTV) concept associated to the customer importance. It is the future net value of a customer via a relationship. The lifetime value is important in order not to over invest in a customer without getting anything back. It is important to identify the LTV of customers. This can be done in both directions in time, historically and in the future. If this is done for the customers, four classes of customers can be identified:

- Low historical value and low future value
- Low historical value and high future value
- High historical value and low future value
- High historical value and high future value

At a glance, this might suggest that the least profitable customers should be decommissioned, this it not always true. If they cost more than they attribute, that fact should corrected. But otherwise use the knowledge to tailor make the service offered. The most valuable customers should be given the most resources. Invest in prolonging and widen the relationship. Different ways of investments can be given, e.g. tailor products and services, offer flexible billing and payments, develop a help desk and to devote the best staff to serve these customers. Widening the relationship is about selling in new concepts and concepts yet not used by the customer. This is believed to be easier than to sell the concept to a new customer. (Kotler, 1999).

Segments are classes of customers with common attributes. A segment should be possible to answer the following questions to be considered relevant:

- measurable - it should be possible to estimate variables e.g. size, growth, buy-in potential
- availability - there must be communication and distribution channels to each segment
- substantial - a segment must be large enough to cover the costs for extra marketing and be profitable to be worth of the extra effort
- realisable - a segment must be realistic. If there are no resources to work on a particular segment, the knowledge of its existence is of no use.

A common dividing in the field of CRM are the four categories: Strategic, Analytical, Operational and Interactive. Strategic CRM is, as already mentioned, about strategy, deciding a strategy on how to strengthen the Customer Relations. The work includes areas like process definitions, channel strategy (how should an organisation act across different sales channels) and change strategy. Analytical CRM is about analysing different aspects of the customers and their behaviours thus identifying segments and trends. Operational CRM is about the every day work within the organisation, normally introducing software support within the organisation, but also about measurement (what information we want to spread across the organisation). Interactive CRM finally is about co-ordinating the different contact areas (channels) to the customer, so that the customer is treated properly no matter how he/she is contacted or contacts the organisation. No matter if the contact is through a web site, a contact centre or a personal sales visit, information should flow freely.

Customer Relationship Management vs CMM

SW-CMM comprises an architecture of five maturity levels for organisations developing software. The maturity levels measure an organisation on an ordinal scale on the attribute maturity: initial, repeatable, defined, managed, and optimising. The idea behind the levels is to propose a number of goals to be fulfilled, together with a set of common features: Commitments, Ability to Perform, Activities Performed, Measurement and Analysis and Verifying Implementation. Activities Performed consists of a number of activities, which should be performed, in a way or another, which implements the process. The other common features assure that the process is institutionalised in the organisation.

Commitments are typically policies, and responsibilities, Ability is resources and training, Measurement and Analysis requires measurement of the processes and Verifying Implementation that the top management, project management and Quality Assurance function reviews and assures that everything is in place.

Level 1, Initial, does not put any restrictions or responsibilities whatsoever on an organisation. That is the level an organisation is by default unless higher. Level 2, Repeatable, introduces some management on the project level, e.g. project planning and tracking, milestones, plans. On Level 2, an organisation is very dependent on the persons involved. Level 3, Defined, is characterised by a common process, defined, documented and used in every project, not as is, but tailored to the project needs according to tailoring guidelines. Level 4, Managed, introduces management by using measurement results, both product and process measurement. Level 5, Optimising, is the final plateau, where software process improvement is the way of life and continuously practised.

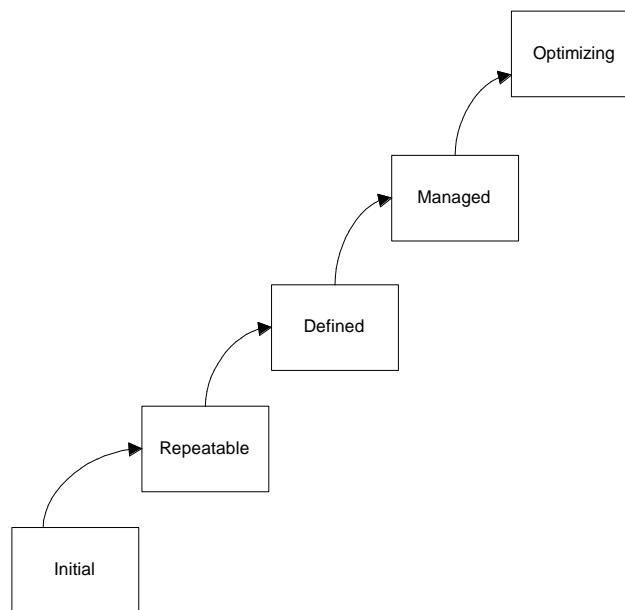


Fig. EKO1. SW-CMM Levels

SW-CMM introduces Goals and Common Features like Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation. The intention with the Activities Performed are to show that the process is *implemented* in the organisation, while the other common features assure that the

process is *institutionalised* in the organisation, i.e. lasts also after the people have gone on holiday. The Commitment to Perform typically prescribes that there should be a *written policy and other prerequisites*, e.g. responsibility. Ability to Perform shows that there are *adequate resources* like knowledge, experience, tools, funding for performing the activities. Measurement and Analysis are assumed to record the *status of the activities*, and Verifying Implementation *assures* that the activities have been conducted, not only a belief but evidence of it.

CMM architecture is based on the assumption that only a stepwise improvement is possible in an organisation, that activities and procedures are dependent on each other, and that some improvements should be introduced together rather than some others. An important assumption in CMM is that there should be a responsible role for each type of process, which is homogeneous in the organisation across different projects. For example, different projects are run differently, requiring different project managers, while SCCB is one role having the responsibility of authorisation across the organisation for all projects. Similarly, SEPG is responsible for the organisation process, which is required not until Level 3. On Level 2, there is no requirement on SEPG as there is no requirement on an organisation process.

CRM Maturity Model

CRM Maturity Model is presented in this section. The underlying assumption is that an organisation should introduce control to the CRM processes by first learning its state-of-the-art by adequate data collection, next actively acting towards a well-defined goals, and finally by presenting a common approach to interaction between the organisation's departments according towards the CRM goals and customer segments. The maturity model is presented in Fig EKO2.

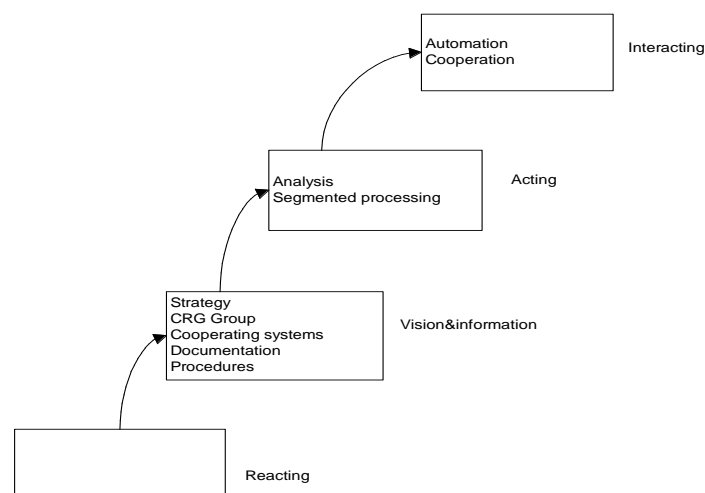


Fig. EKO 2. CRM Levels

Level 1, *Reacting*, is characterised by the lack of documented procedures. At the time of

a crisis, the working procedures are adapted ad hoc to the situation at hand by the actor. There are no documents describing or prescribing the way of work, and thereby the acting is based on the attitude and skills of the actor, leading to different acts even in same situation. This is very similar to CMM Level 1 characterisation.

Level 2, *Vision and information*, is characterised by data collection, a common view on the customers in the organisation and the costs associated to activities, including a CRM strategy as a common denominator. In other words, the organisation learns to know its state-of-the-art, what activities are performed and what the costs of the activities are. The data collection lays a ground for the next level.

Level 3, *Acting*, is characterised by a segmented marketing and advertisement. The customers are classified by type and profitability. The activities are adapted to the customer category. The business is continuously analysed and modelled, based on the data collected on the Level 2.

Level 4, *Interacting*, is characterised by co-ordination over different departments or companies or business areas, customer focused alliances and acting as one organisation towards the customers. The activities performed are directed towards the goals. There is knowledge of what the customer wants, and a holistic view on customers. A mutual value creation is practised. The roles of departments within a company are effectively determined and the work is supported by automatic flows/processes.

In the CRM Maturity Model, Goals, Prerequisites, Activities and Verification are included. The common feature Prerequisites merges the CMM common features Commitment to Perform and Ability to Perform. The common feature Verification merges CMM Measurement and Analysis and Verifying Implementation.

In the next three sections, the different maturity levels are described more in detail including their key process areas (KPA), and roles and responsibilities.

Level 2 Vision & Information

Level 2 is characterised by activities towards obtaining knowledge of the customers, customer relationship management, and its associated costs. There are five KPAs on this level:

- The Strategy
- CRM Group
- Co-operating systems
- Documentation
- Procedures

The Roles

Each of the KPAs introduces a particular role, usually a group by name. There are the following groups: the strategy group (SRG) introduced by the Strategy KPA, CRM

Group (CRMG) introduced by the KPA CRM Group, a group for co-operating systems (CSG) introduced by the Co-operating Systems KPA, a group for Documentation (DG) introduced by the Documentation KPA, and a group for procedures (PG) introduced by the KPA Procedures.

The Group members

Each group involves a chairman and a project manager (normally one and the same person) for the activities to be performed by the group in case. The work is performed on a project basis. Normally the project is both introduced and terminated by the CRMG. In addition to that, each group also involves a role responsible for documentation of the work. The documents are used for measurement of the activities as well as being a part of the normal organisational learning. SRG and CRMG also involve a role of business responsible as the work might introduce changes in the organisations business process. CRMG also involves a role of purchase responsible, and CSG involves information owners and system owners.

A person can be a member of more than one group and hold more than one responsibility within a single group, all dependent on the size of the project. As this is a generic model, no consideration has been taken to existing policies and work within an actual organisation. It is always the CRMG:s responsibility to adjust the model into the actual surroundings.

The Key Process Areas

All the KPAs include Goals, Prerequisites, Activities and Verification. The Prerequisites typically are resources, management commitment, and knowledge and experience, while Verification includes measurement and reviews of the activities, meetings for tracking issues to closure, periodically or on event-driven basis. For space reasons, only the goals and activities will be described in this paper. The full details can be found in (Ekstam, Karlsson, 2001).

The Strategy

Basic assumptions about the work in KPA

A CRM-project needs a strategy to become successful. This has been emphasised by appointing a special KPA to this subject. Normally strategy is considered being a responsibility for the top management (Bolman et al 1997). And they are here given the ability to do so and then delegate the operational work to other people (the CRMG).

Goal 1 There exists a written document including the objectives of the CRM investment.

Goal 2 There exists a strategy for the fulfilment of the objectives.

Activities of SRG are to

- determine (assesses) the current state
- define and document the goals,
- communicate them to the employees.
- define a strategy for the fulfilment of the objectives.

The assessment should answer questions like "What interfaces do we have with our customers?", "What processes do we have and what is conducted in the processes?", in other words what do we do currently for our customers through which interfaces.

CRM Group

Basic assumptions about the work in KPA

The CRM group is the core of the CRM process. They drive the work within the organisation, co-ordinating different departments. It is their responsibility to staff and carry through all other KPA:s.

- Goal 1* The CRM efforts of the organisation are co-ordinated.
Goal 2 CRM Group (CRMG) is the interface of the organisation regarding CRM issues.

Activities of CRMG are to

- decide the priorities of the CRM efforts,
- define a schedule with estimated effort and time for the activities in each KPA
- track and revise the schedule continuously
- initiate, decide and track all KPAs
- inform the employees on a periodic basis.

Co-operating Systems

Basic assumptions about the work in KPA

In order to get a clear picture of the customer, data from various sources needs to be aggregated. Also in order to gain efficiency in the work process, automation might be needed. These tasks require different Information systems within the organisation to exchange data. Data exchange is difficult and requires attention in order to work well.

- Goal 1* Data recording is integrated.
Goal 2 The systems related with customer management share common data.

Activities CSG are to

- decided which systems should co-operate together with CRMG
- develop a plan for how systems should co-operate
- meet the information owners periodically and on event-driven basis
- track the co-operation based on the plan for co-operation

The meetings are intended to answer questions like "What should be integrated, how should the information be integrated"

Documentation

Basic assumptions about the work in KPA

Getting a clear picture of the customers are not only a question of integrating data from different computer systems also the employees must start to document their work, especially about customers. Documenting the activities performed is an important part in

creating knowledge. Knowledge is bound to become an ever more important competitive advantage (Davenport 1998)

Goal 1 The activities in the organisation are documented.

Goal 2 Every employee documents periodically the tasks completed.

Activities of DG are to

- develop a documentation plan for what should be documented, and how, e.g. country, other media, format and layout
- determine the information owners.

Processes

Basic assumptions about the work in KPA

In order to increase the return of investment it is important to work smarter, not harder. In order to assure that the customers are treated correct and consistent in an effective way the organisational processes are gone through.

Goal 1 The activities in the organisation are documented.

Goal 2 The activities are conducted in a structured way.

Activities of PG are to

- document and analyse the current activities
- define new activities
- train the affected individuals in the new procedures.

Level 3 Acting

On this level, the customers are classified into different segments and the activities are focussed according to the segment. On this level, the work is characterised as acting instead of reacting. The work is directed towards a situation with knowledge of the customer wish and the acting is directed towards to giving the customers exactly that.

There are two KPAs on this level:

- Analysis
- Segmented processing

The groups responsible for the KPAs are Analysis Group (ARG) and Segmented Work Group (SWG)

Analysis

Basic assumptions about the work in KPA

In order for the organisation to make rational decisions it is important that there exist a comprehensible model of the business. It is important that decisions are based in some sort of analysis. To assure that decisions concerning customers are good, a model of the customers and the business is created.

Goal 1 The organisation has a model of the business, i.e. knowledge of the customers and how they act, and how they interact with the customer.

Activities of ARG are

- analyse data
- develop a theoretical model of the customer portfolio, dividing customers in segments according to predefined criteria.
- track the model periodically
- change/adapt the model as needed

Adequate support for the ARG could be provided by a DSS or by a OLAP-tool

Segmented Work

Creating segments of customers are done in order to find similarities between different customers. Experience with one customer might than be used for another one. The profit a customer attributes to the organisation should affect the resources spent. On this maturity level the organisation has the skills and knowledge needed to work with segmentation in an effective manner.

Goal 1 The customer activities are derived from the segments.

Activities of SWG are to

- define a plan of work
- communicate the plan to the employees.
- revise the plan on a periodic basis.

Level 4 Interacting

On this level, there is data collected in the organisation according to the earlier levels. Learning is based on the facts about the customers, to adapt the service to the customers' needs and the profit potential. To be able to give better service to the customers, it is important to act over several companies or business areas both for dissemination of knowledge of the customer and to conduct common activities to solve problems for customers.

To increase the internal efficiency it is now initiated some automation. The processes are studied and redundant work is minimised. Activities are conducted automatically if possible, for example that a certain activity is started when a customer shows a particular pattern.

There are two KPAs on this level:

- Automation
- Co-operation

Automation

Basic assumptions about the work in KPA

Here a holistic perspective on automation is taken. Earlier, automatic processes have been used only to solve problems within different KPAs, while here the work is centred around different roles for the employees. The purpose is that the employees should have different, well-defined roles. That means studies of the processes related to each other and to associate these to roles. This is required to see what processes can be automated. Automation implies two things, both the mechanisation of processes and the introduction of self-service in order to get the customers to carry out more work themselves.

- Goal 1* Each process is related to a role.
Goal 2 Each employee possesses one or several roles.
Goal 3 Resources are associated to roles.
Goal 4 Activities are automated if financially and technically realistic.

Activities of AG are to

- appoint responsible for each segment to increase the loyalty,
- determine the degree of service and the type of marketing to the segment.
- study the processes within the organisation
- associates the processes to the roles
- study the potential profit with automating parts of the business
- revise the role distribution if needed.
- each employee gets one or several roles.

Co-operation

Basic assumptions about the work in KPA

Normally a business environment is not static, Organisations merges, divides and changes, Even if the organisation is not static, it is important to keep treating the customers in a consistent way. This KPA deals with the process of integrating two organisations (i.e. business, subsidiaries or departments). An important prerequisite is that both merging parties has worked according to this model.

- Goal 1* Companies exchange customer data.
Goal 2 Companies conduct a common work of common customers.
Goal 3 Companies have a common model of the customers and business with them.

Activities of CG are to

- meets periodically and on event-driven basis
- study all the KPAs on all levels to get a common model, super model, which gives the common view derived from co-operation. In that, only the common customers are included.

Concluding Remarks

A maturity model for CRM has been briefly presented. The work has been conducted by observing the world in the context of a medium sized company, by studying literature about CRM and process improvement, and by creating the model.

Clearly, the model is only a proposal, to be validated by empirical studies in an industrial setting. The work is in progress, but more studies are needed. Interested companies are welcome to obtain the complete model from the authors, against feedback.

References

- Bolman G. Deal, T., "Reframing Organisations" John Wiley and Sons Ltd, 1997.
- Buttle, F., 2000, "The S.C.O.P.E. of Customer Relationship Management", Available 20 June 2000 www.crm-forum.com.
- CapGemini Ernst & Young, "CRM Core Presentation" available 2001-05-03 at www.cgey.com
- Davenport, T. et al. "Working Knowledge", Harvard Business School Press, 1998.
- Ekstam, H., Karlsson, D., "Modell för CRM i organisationer", Thesis 2001-X-130 The department of Computer and Systems Science, the Royal Institute of Technology Stockholm, 2001.
- Karlöf, B., "Strategi i verkligheten", Ekerlids Förlag, 1996.
- Kotler, Ph. "Marketing management" Prentice Hall International, 1999.
- Paulk M.C. et al., "The capability Maturity Model – Guidelines for improving the Software Process", 1995.

Authors and Companies

MEng Henrik Ekstam

Undergraduate Student Henrik Ekstam has studied Computer Science at the Royal Institute of Technology, and Economics at Stockholm University. He currently has an assignment as CRM specialist at Bergman & Beving Tools (BBT), where he is also working on issues of IT-strategy.

MEng Daniel Karlsson

Daniel Karlsson is a undergraduate student who has studied Computer Science at the Royal Institute of Technology. Daniel is currently working at Bergman &

Beving Tools (BBT) as a CRM specialist. He has been working with IT-strategy and developed an IT-strategy and policy for BBT.

Professor Terttu Orci

Professor Terttu Orci received her PhD at the Royal Institute of Technology (KTH), Stockholm, and is currently appointed as a professor at Stockholm University in Computer and Systems Sciences. She works as lecturer, researcher, and supervisor at Stockholm University/KTH. She is in charge of the competency track in Software engineering in the education programs at both universities, with special focus on software process improvement and software metrics. She is currently working with development of a capability maturity model for small and very small organisations. Although the current research interest is in software engineering, the prior research interests include databases, temporal aspects of database modelling, and automated theorem proving and AI. She has several times been appointed by the EC as external expert in evaluations of project proposals and in projects. She also works as consultant in industry in various co-operation project between academia and industry. External activities include work in SweSMA, Swedish Software Metrics Association, a national member of the European FESMA and EuroSPI.

Bergman & Beving Tools

Bergman & Beving Tools (BBT) is a Swedish whole seller in the Industrial Sector. Employing some 800 co-workers in the Baltic area. Through different subsidiaries and e-commerce BBT sells equipment and supplies for manufacturing industries and construction companies. BBT has been the initiator of the thesis that founded the described model.

The department of Computer and Systems Science

The department of Computer and Systems Sciences is a department belonging to two universities, to Stockholm University and the Royal Institute of Technology (KTH). The number of employees is 150, including six full professors, and the students including 2000 undergraduate and 60 postgraduate students. The research areas at the department include information systems, informatics, computer science, software engineering, artificial intelligence, and computer security.

Experiences On Outsourcing Requirements Specifications

Friedemann Kiedaisch, Martin Pohl, Joachim Weisbrod

*DaimlerChrysler AG
Research and Technology*

Siegfried Bauer, Stefan Ortmann

*DaimlerChrysler AG
Passenger Car Development*

Introduction

With the increasing importance and complexity of software systems in cars, the car manufacturers are faced with new challenges: New architectures must be invented and the development must be accompanied by completely new processes. And all this under the pressure to reduce time-to-market and development costs. In this paper we focus on the early phase of software development and describe a successful approach taken at Mercedes-Benz Technology Center (MTC) to obtain high quality and well structured requirement specifications while minimizing the extra workload of the development teams. The general design decisions will be presented. We derived a two phased specification process. While in the first phase the raw specification of the features was done by experienced requirements analysts in close collaboration with the domain experts, in the second phase partly inexperienced staff was hired to detail the specifications exploiting the various sources of information available. The details of our approach and the conditions under which the approach promises to work satisfactorily will be discussed.

Requirements Engineering at DaimlerChrysler

It is a well accepted fact, that expenses and risks for software development can be decreased by spending more effort in the early stages, especially in the requirements specification phase. The cost of resolving mistakes detected in this stage is by a factor of up to 1,000 smaller in comparison to resolving them in later phases [1]. What makes things worse is that nowadays often the major proportion of errors emerge from requirements elicitation and specification. In this context we made the following three

observations: (i) Most publications and textbooks dealing with requirements engineering (RE) – e.g. [3, 4, 5]– assume that systems are being built from scratch. Specifying from scratch means, that we need to trawl for, negotiate, prioritize, and finally specify requirements for a completely new system. However, in many business areas systems are built in an incremental fashion, i.e. the $(n+1)$ th generation of the product basically inherits the features of the n th generation, only enhancing the system with more or less additional innovations. So we need to cope with a situation, where we can (and maybe should) exploit major parts of specifications written for predecessor products. (ii) The potential reuse of specifications for predecessor versions raises the next problem: Requirements are often not very systematically dealt with. Scope creep and changes are not completely incorporated into the specification document. Therefore the requirement specifications do not cover the complete functionality of the final product. We even find complex and challenging systems being built without a rigorous specification of the requirements. In the automotive world, for instance, such an approach may work very well with long term suppliers, who are very competent and in many cases know better than the customer what to develop. (iii) The workload of the development engineers is often very high and usually they are not able to spend much time for the detailed elaboration of requirements specifications (and therefore deliberately delegate this task to the supplier).

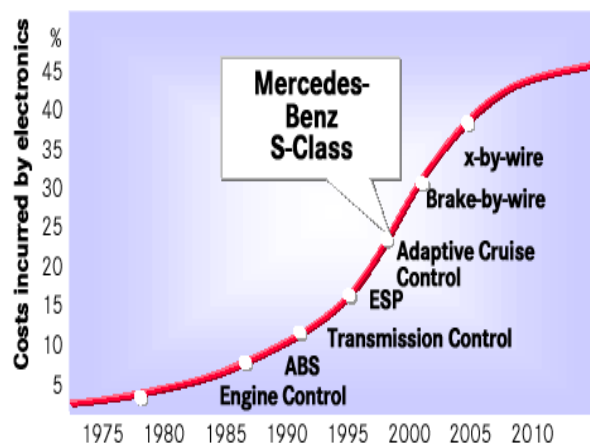


Fig.FK.1 Increasing importance of software (as vital part of electronics) in automotive industry

In automotive industry software has become more and more important in the past and it is expected that the importance will increase further (Fig.FK.1). Thus a rigorous specification of requirements of the features will become as well more important. Furthermore, software related features offer the car manufacturers the key to differ from their competitors. Consequently, it is an important goal of the automotive industry to drastically reduce the dependency from individual suppliers delivering software based parts like embedded systems. Software development of vital features is planned to become an in-house activity. To reach this goal, two steps are important: (i) The software development must be separated from the development of the hardware. (ii) Since the features are being built in parallel by two organizations (the software part in-house and the hardware by the supplier), the corresponding specifications must be as complete as possible from the very beginning. Changes to the specifications must be incorporated and

tracked in a systematic way to keep the software specification consistent with the hardware specification.

Now the basic question is, how to obtain these high quality specification documents in a situation as described above, where engineers with domain expertise are faced with a lot of new technical challenges and overloaded with tasks and deadlines. Furthermore, those core development teams are often not familiar with the techniques to be applied in systematic requirements engineering. This means, they would not only have to accept an additional task (“build those high quality requirements documents”), but also an additional training on how to do so (“how to specify high quality requirements documents”).

In this paper we describe a successful approach taken at Mercedes-Benz Technology Center (MTC), how high quality requirements specifications can be obtained while minimizing the extra workload of the core development teams. The basic idea is to assign and hire supplementary staff for doing the job.

We will first motivate and describe the way we tackled this situation (section “Outline of the process”). Section “Experience and evaluation” discusses and evaluates both the process and the results achieved in the given project. Concluding remarks and some hints on future tasks and opportunities are illustrated in the final section “Conclusions”.

Outline of the process

First, we introduce the **project context**: The system to be specified was the set of embedded systems responsible for driver and passenger comfort. The focus of the specification process was on software requirements. The corresponding functionalities include basic (but nevertheless surprisingly complex) features like in car illumination or wing mirror control as well as more innovative and complex features the interested reader should look for in future Mercedes-Benz models. The domain knowledge for these different functionalities is distributed amongst several departments at MTC. The complete behavior to be specified was clustered into some 70 features. The requirements specifications for these features were to be completed within 6 months, the overall effort was about 100 person months.

The following **main goal** and **main constraints** were fundamental for the overall process we established in order to accomplish this task:

Goal: Deliver high quality and well structured requirements specification documents.

Constraints: (1) Deliver those documents fast, but do spare the very scarce resources of the corresponding domain experts. (2) Exploit the fact that lot of information to be used for this specification is already available in all sort of documents from previous car models.

From these preconditions we derived the following **general design decisions**:

- Provide a thorough and well documented template for requirements specification to make sure everyone knows exactly what to do and why to do it.
- Provide a well structured specification template that supports not only implementation but also reuse of requirements specifications in future development cycles of future car models.
- Establish a highly distributed process in order to have much work done in parallel.
- Synchronize distributed activities as often as necessary – but no more.
- Structure the process in a way to keep the balance between simplicity and flexibility on one hand and efficiency and stability on the other.
- Foster formal and informal communication between the members of the specification team: We are running a new process and therefore we are sure that we can improve

lots of things under way.

- Try to approach the car development experts in a goal oriented, systematic, and well prepared way in order to ensure efficient use of their time.
- Try to get an advantage from dealing with all different sorts of people and all different sorts of expertise.
- The use of a requirements management tool is highly recommended in later phases of development. But this does not mean that every member of the specification team needs to get used to this tool. Provide a automatic migration from word processing templates to a requirements management (RM) tool instead.

From the project context and this set of goals, constraints, and design decisions we derived a two phased **specification process**:

1. raw specification (document & consolidate scenarios)
2. fine specification (document detailed requirements & peer reviews)

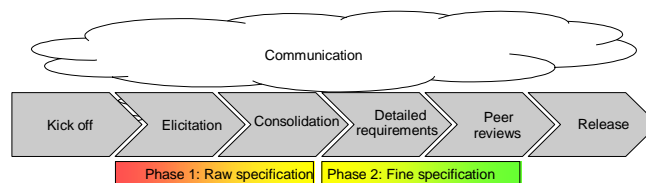


Fig.FK.2: General steps of the specification process.

The two phases of the process can be characterized as follows (see Fig.FK.2):

Phase 1: Raw Specification.	
<i>Goal</i>	Identification and documentation of coarse grained overall system functionality and interaction.
<i>Constraints</i>	Domain experts should be contacted as seldom as possible. Information should be reused wherever possible.
<i>Inputs</i>	Interviews with domain experts, all sorts of existing documentation (requirements specifications, user documentation, minutes, presentations, ...).
<i>Outputs</i>	For each feature: specification of the basic functionality, scenarios, and the interfaces to interdependent features; overall software system design: clustering of features and their interdependence (interface design).

Major parts of the system have already been developed for existing products, there are some innovative completely new features, though, and many updated features are enhanced in one way or another.

Phase 2: Fine Specification.	
<i>Goal</i>	Detailed and unambiguous requirements specification enabling SW development and potential reuse in future development cycles; consistency of features on system level.
<i>Constraints</i>	Further consultations with domain masters only on demand; integration of additional requirements analysts.
<i>Inputs</i>	For each feature: specification of the basic functionality, the scenarios and the interfaces to interdependent features.
<i>Outputs</i>	For each feature: complete requirements specification according to predefined template with stable and consistent feature-to-feature interfaces.

Parallel to these phases we explicitly installed an important ongoing activity, namely

Communication.	
<i>Goal</i>	Ensure and foster formal and informal, regular and sporadic communication between all persons involved in order to exploit their different backgrounds, experiences and

<i>Constraints</i>	individual talents. Not all involved people working in one place.
<i>Inputs</i>	Conclusions drawn from random document inspections, intermediate results, feedback from all sorts of roles involved, actual problems and hints
<i>Outputs</i>	Improvements and optimizations in process, templates, guidelines and understanding.

The following **main roles** were involved in the specification process:

Domain expert: MTC development engineer with domain knowledge but little time for requirements specification; each domain expert is officially responsible for one or several of the features to be specified.

Senior requirements analyst: member of the specification team with some domain expertise; typically, these persons were MTC members that were internally assigned to work on requirements specifications.

Requirements analyst: member of the specification team without domain expertise; typically, these persons were external consultants hired for delivering requirements specifications.

Change management: responsible for keeping the set of requirements specifications consistent, which in particular means to keep track of the interfaces between interdependent features.

Quality management: responsible for quality assurance and quality management.

RE expert: observes the process and its intermediate results in order to identify and initiate possible improvements; consults the specification team.

In a nutshell, the job for each member of the specification team (senior requirements analysts and requirements analysts) was to fill in a predefined requirements specification template by trawling for and analyzing the information needed in working documents like related specifications, minutes, user documentation and other types of documents. There was of course communication with the corresponding domain experts.

In the rest of this section we will more closely illuminate both phase 1 and phase 2 as well as the feature requirements specification template.

Phase 1: Raw Specification

The implicit challenge of this phase is to bring together different people with different backgrounds and different expertise. This is why this phase consists of preparing and imparting steps on the one hand and communicative steps on the other hand.

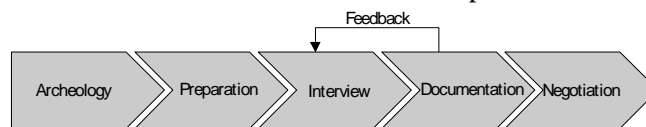


Fig.FK.3: Steps of the raw specification phase.

Archeology.	
<i>Goal</i>	Senior requirements analyst basically understands the feature he is assigned to as well as the overall system.
<i>Constraints</i>	Information available from predecessors, but possibly distributed among heterogeneous sources
<i>Inputs</i>	Specification documents of related existing systems and all sorts of additional documents.
<i>Outputs</i>	Feature context diagram.
<i>Roles involved</i>	Senior requirements analyst.

In this step, the senior requirements analyst read all available papers to get used to the feature domain and to have an overview over related existing systems and documents.

Preparation.	
<i>Goal</i>	Senior requirements analyst is prepared to talk to the domain expert with sufficient

<i>Constraints</i>	knowledge of the subject.
<i>Inputs</i>	--
<i>Outputs</i>	Feature requirements specification template, feature context diagram.
<i>Roles involved</i>	Senior requirements analyst, RE expert.

This step has two aspects. With respect to content, the senior requirements analyst collected shortcomings and knowledge gaps, where information cannot be extracted from the available documentation. But he also had to take into account that the domain expert may be reluctant or wants to get convinced of this repartition of work. Thus it had to be carefully considered how to structure and word the questions.

Interview.	
<i>Goal</i>	Get a common understanding of the feature and the required functionality, as well as clarify open questions, misunderstandings, and differences to former systems.
<i>Constraints</i>	Willing and qualified domain expert.
<i>Inputs</i>	Domain specific questionnaire.
<i>Outputs</i>	Interview minutes.
<i>Roles involved</i>	Senior requirements analyst, domain expert.

Documentation.	
<i>Goal</i>	Raw characterization of the feature.
<i>Constraints</i>	--
<i>Inputs</i>	Domain expert interview minutes, feature context diagram, all sorts of related documentation.
<i>Outputs</i>	Feature scenarios (part of feature requirements specification).
<i>Roles involved</i>	Senior requirements analyst, RE expert.

Feedback.	
<i>Goal</i>	Ensure common understanding and completeness as well as consistent use of template.
<i>Constraints</i>	--
<i>Inputs</i>	Intermediate feature requirements specification.
<i>Outputs</i>	Improved feature requirements specification.
<i>Roles involved</i>	Senior requirements analyst, domain expert.

Synchronization/negotiation of interfaces.	
<i>Goal</i>	Clear responsibility and interdependence of separate features, ensure consistency of overall system.
<i>Constraints</i>	Sufficiently complete and stable documentation of each feature.
<i>Inputs</i>	Feature requirements specifications.
<i>Outputs</i>	Overall system diagram.
<i>Roles involved</i>	Senior requirements analyst, change management.

Phase 2: Fine Specification

The approach in the second phase of the specification process was similar to the first phase. However, since the functionality of the modules was already defined on a high level through scenarios and previous specification documents, the main emphasis was now put on refinement and completion.

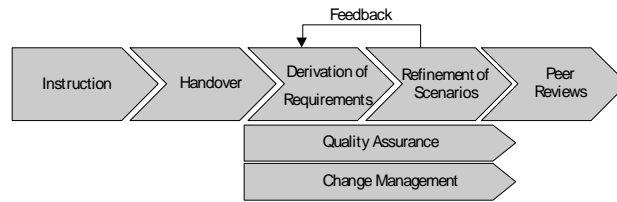


Fig.FK.4: Steps of the fine specification phase.

As first and important step there was an instruction phase for the new requirements analysts. There were theoretical trainings on requirements engineering in general as well as on the actual process, but also hands on exercises and practical examples.

Handover of documents.	
<i>Goal</i>	Brief requirements analyst wrt. state of specification document, sources of information used, short term todos, and open points
<i>Constraints</i>	--
<i>Inputs</i>	Intermediate specification documents.
<i>Outputs</i>	Intermediate specification documents.
<i>Roles involved</i>	Senior requirements analyst, requirements analyst.

The feature specification documents were assigned and handed over from the senior requirements analyst to the requirements analyst. The requirements analyst was instructed about open points in the working documents.

Derivation of requirements.	
<i>Goal</i>	Derivation of functional and non-functional requirements from the scenarios and their detailed and unambiguous specification.
<i>Constraints</i>	Further interviews with domain experts should be the exception.
<i>Inputs</i>	Interview minutes, description of the basic functionality and scenarios, all sorts of related documents.
<i>Outputs</i>	Functional and non-functional requirements.
<i>Roles involved</i>	Requirements analyst.

Refinement of scenarios.	
<i>Goal</i>	Improve scenarios.
<i>Constraints</i>	--
<i>Inputs</i>	Scenarios and requirements.
<i>Outputs</i>	Refined scenarios.
<i>Roles involved</i>	Requirements analyst.

During the detailed analysis of the feature needed to derive the requirements, the requirements analyst sometimes identified overlooked functionality, inconsistencies, and inappropriate scenarios.

Quality assurance.	
<i>Goal</i>	Improvement of common understanding of specification template and recipe and formal consistency of specification documents.
<i>Constraints</i>	Due to time constraints only sampling was possible.
<i>Inputs</i>	Intermediate requirement specifications.
<i>Outputs</i>	Inspection records.
<i>Roles involved</i>	Quality management, RE expert.

Parallel to requirements specification quality management continuously inspected selected documents and reported conspicuous discoveries in the weekly meetings. As quality management has no deep domain knowledge, the focus of this step was on formal aspects.

Peer review.	
<i>Goal</i>	Improve quality of requirement specification, enhance common understanding of content of the documents, eliminate open points, mistakes and ambiguities.
<i>Constraints</i>	Only requirements analysts involved in the specification process were involved into the review activities.
<i>Inputs</i>	Requirement specifications.
<i>Outputs</i>	Inspection records, improved requirements specifications.
<i>Roles involved</i>	Quality management, requirements analyst.

All requirements specifications were inspected in a formal inspection process by two other requirements analysts.

Change management.	
<i>Goal</i>	Ensure consistency, especially of interfaces.
<i>Constraints</i>	--
<i>Inputs</i>	Change requests.
<i>Outputs</i>	Adapted overall system design, adapted feature requirements specifications.
<i>Roles involved</i>	Change management, requirements analyst.

Suggested changes of interfaces were reported to the change management and the relevant requirements analyst(s). Passed changes were documented and the affected feature requirements specification were adapted.

The Feature Requirements Specification Template

The starting point of the specification was a collection of heterogeneous documents, written by people of several departments, which often emphasized more the technical and hardware aspects than the software. Furthermore, many sources described solutions instead of actual requirements. In order to avoid all these defects in the new specifications, we defined a feature requirements specification template to capture and document the results of the process. The structure of this template follows strongly the suggestion in [7].

In order to facilitate the use of the template, the template had to reflect the process, and the filling order had to correspond to the process steps. So we defined a structure providing separate sections for the results of raw specification and of fine specification, while [7] unites them in the same chapter of his structure:

- basic feature description
- context diagram
- scenarios
- detailed requirements
- interface and data descriptions

The last item was emphasized as a section on its own in order to support the planned approached for the following development steps [2] that need this input.

Our goal was to produce a standardized, unified, well structured, and reusable documentation, which contains delimited, uniquely identified requirements that easily can be referenced. Finally, all requirements should be stored in a central repository. Actually, in the first step we defined a complete data model for a requirements management database (requirements management information model or RMI, for short). However, given the tight project schedule it did not seem appropriate to have the requirements analysts learn to use new tools for requirements management. Furthermore, we had to take care of the risk that our activities fail.

Taking into account these conditions, we chose to use a word-template for requirements

specification, because MS-Word is the most common tool to both, domain experts and requirements analysts. This way, they had nothing to learn and the fallback solution to a “standard” specification document was for free. To achieve the goals given above we stipulated the use of tables with given rows and columns. We provided only few small sections of plain text for introduction and overview, which rarely must be adapted. For the purpose of references we introduced a simple numbering scheme.

In addition to the overall structure of the template, we introduced subsections to unite contents of the same structure. This, in combination with the use of tables made the automatic import of the documents into a database possible. Finally, this database serves as central repository for the specifications.

Because we planned to use special features of the database tool later on, we prepared their installation by special attributes in our table pattern. For instance, we inserted a reference field, which should be filled with elements of the numbering scheme, to provide traceability by means of links in the database, which could be extracted automatically from corresponding attribute values. Another example is the author/source-change date-field to provide history features.

The aim of choosing another couple of attributes was to take the writer by the hand and give hints, which information in which detail he has to look for, especially to mark important pieces of information, e.g. input/output/trigger-fields, which were necessary for interface design.

As a result of all these goals, aims, additions and improvements, our template consisted of six different types of tables, each table built of three to eleven types of cells. Since such a template is neither self explaining nor the place to provide adequate information on why and how to fill in the cells, we decided to supply an extensive **guideline**, called the “recipe”, which helped the writer to fill in the template in the designated manner. The recipe contains explanations about which parts of information belongs to which field, sections to motivate why some attributes actually exist, general recommendations about filling orders and how to simply summarize or reuse frequently occurring patterns.

Experience and evaluation

The requirements specifications

The activities described in section “Outline of the process” resulted in about 50 feature requirements specifications. Some additional 20 features were not completed for several reasons like time delays, changing responsibilities, or general technical problems. Obviously, none of these reasons is related to our requirements specification process. On average, each specification document covers about 40 pages. 48% of the specifications delivered are expected to be highly stable, 24% are still ranked as quite stable (Fig.FK.5).

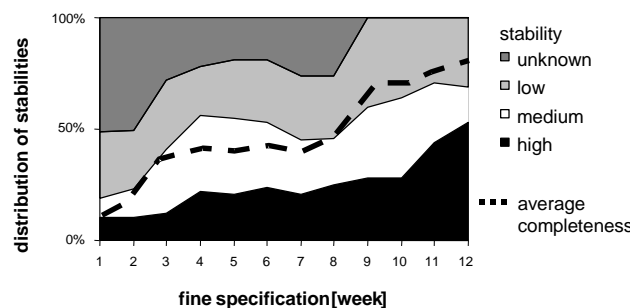


Fig.FK.5 Evolution of stabilities and average completeness during fine specification

Altogether, these figures represent a quite satisfying result that is regarded as being very useful. There are some special characteristics in the given project that helped to achieve this success:

The requirements specification process outlined in section “Outline of the process” was applied for the functionality in the car interior. This functionality is basically evolved from previous models or other types of cars developed by Mercedes-Benz. Hence, most of the requirements could be extracted directly from other specifications (best case) or the information needed to document the requirements was available but distributed over different kinds of documents and people (normal case). In most cases, enhancements or added functionality to such features could also be handled well in this setting. Highly innovative or completely new features were a more challenging task. Generally, requirements analysts without domain expertise cannot specify such features without constant feedback and support by the domain experts. Nevertheless, even in these cases the work load of the domain experts was dramatically decreased by means of the specification work done by the requirements analysts.

The People

The overall process involved some 30 domain experts (each of them responsible for 1 to 6 features), 10 senior requirements analysts, 13 requirements analysts, and three additional persons responsible for change management, for quality management, for general RE matters, as well as for the requirements management tool (Fig.FK.6).

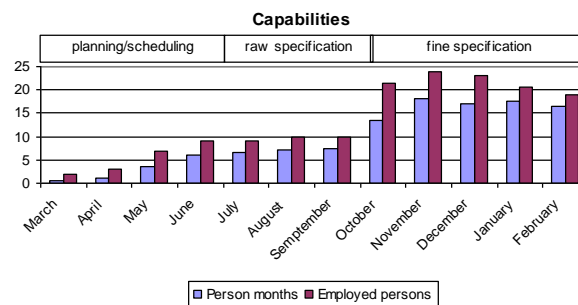


Fig.FK.6 Overall resources bound to requirements specification (both workload and persons)

Obviously, the domain experts played a crucial role for the overall success. Their support and cooperation was needed in both phases of the process. There is a bunch of possible reasons for a domain expert for being not cooperative like for instance personal fear about losing individual influence, politics, time pressure, lack of understanding, lack of expertise. So the domain experts had to be convinced of the opportunities of the approach instead of only seeing the potential risks of making their implicit knowledge explicit.

From this consideration we directly derive the observation, that social skills of (senior) requirements analysts represent a major success factor for the overall process. Furthermore, we expect an excellent (senior) requirements analysts to have good analytical skills as well as good writing skills. The good news about this need for different types of skills is, that a decent mixture of persons with different strong and weak points seems sufficient, as we will point out in the following:

The requirements analysts employed in the second phase of the process had different backgrounds, different skills and different views on their task. Few members had experience in requirements engineering, others in general technical documentation; some

had already worked for electric/electronics in passenger car development before, some were complete newcomers to this application domain. Last but not least, some members were working at the MTC building next to the car developing domain experts, whereas a second group was located completely off-site without direct and informal access to these experts.

An important advantage of the different types of people involved in the fine specification phase was, that they could benefit from each other. This synergy was supported by the formal and informal communication. Furthermore, this circumstance, which had been regarded as a risk at the beginning of the project, turned out to be of advantage for the specification process. The requirements analysts were “naive” enough to have an unbiased view on the features. During the derivation of the functional requirements, they questioned the scenarios defined in the first phase of the process, which often led to a refinement of the scenarios and to the resolution of ambiguities.

Our final message in this context is quite obvious, but nevertheless important: there are different types of domain experts, different types of domains, different types of features, as well as different types of requirements analysts. It is worthwhile to spend some time thinking about which feature to assign to which analyst. Some combinations work, while others are bound to fail.

The Process

We divided the process into two phases with different roles involved. In the first phase the raw specifications were documented by the senior requirements analysts with the support of the domain experts. Additional requirements analysts were employed in the second phase. Their job was to derive and refine the functional requirements from the scenarios documented in the raw specification. The clear advantage was, that experienced coworkers started with the preparatory work of raw specification (including the clustering of the features). Only on this basis the specification by an inexperienced requirements analyst became possible. However, a good synchronization between the two phases was necessary.

The raw specification phase took about 3.5 months of time and about 25 person months of effort covered by 10 individuals. The fine specification phase took also about 3.5 months of time and about 75 person months of effort covered by 22 individuals. The overall effort for the domain experts was not recorded and varied from feature to feature. The main interview in the raw specification phase, however, took no more than one hour, on average (see Fig.FK.7). Extrapolating this analysis the domain experts’ overall effort was less than one person month overall.

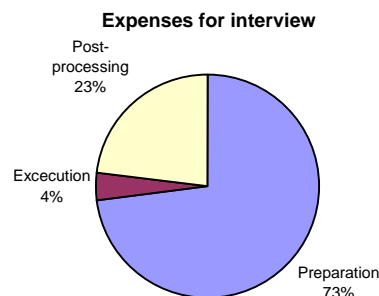


Fig.FK.7: Distribution of workload for phase 1 interviews. Domain experts participation is restricted to “Execution”.

Our RE process started with the overall functionality already clustered into features. On one hand this was regarded as a drawback, because it obviously anticipates design decisions. On the other hand it enabled the assignment of features to individual requirements analysts and was the precondition for a distributed and therewith fast specification. From our point, however, it is an open point, whether there should be a different approach for mature and well known features on one side and for innovative features on the other side.

The progress of the work was continuously tracked by the quality management team. For each feature the requirements analyst rated both stability and completeness with respect to several aspects. In order to check and harmonize these ratings, we constantly inspected particular feature requirements specifications that were systematically selected by the criteria ownership, stability and completeness. Furthermore, with these inspections we assured the formal quality of the documents. The findings identified were communicated and discussed in the regular meetings. Especially this measure enhanced the common understanding of form and content of the specification template.

The final and critical step of our process is the final transfer of the feature requirements specification from the requirements analysts to the responsible domain expert. Our experiences discourage from making this transfer a formal act, such as having the domain expert officially sign the completeness and correctness of the specification. Quite naturally, people do not like to officially accept responsibility for specifications fixed by newcomers in an innovative, unfamiliar way. We suggest to completely rely on the value and usefulness of the delivered documents: As long as the domain experts were informed on the progress of the specification in a constant manner, they deliberately adopt the complete feature specification document, finally.

The Template

In order to standardize the requirements specifications the analysts were provided with a requirements specification template. To ensure the uniform use of the template, we additionally provided a detailed recipe how to use the template. Therefore not much practice was necessary, before the (senior) requirements analysts could start with the specification of the features. For the same reason a standard word processing program was used instead of a more exotic requirements engineering tool. The recipe became one of the most important meta-documents within the process. But to gain the optimal benefit, such a document has to be regularly (say weekly) updated, to account for misunderstandings or parts, which made the writers the most serious problems.

As expected, the first version of the template was not perfect. Minor changes and additions to the template, however, caused tiresome and often stupid rework for the analysts, as given the use of MS Word every single specification document had to be manually adjusted. This manual rework noticeably reduced the savings gained from not introducing a RM tool. In future similar projects, however, we will now start with a better template from the very beginning. The final migration from Word template to RM tool (DOORS) worked quite smoothly, and the corresponding requirements management information model (RMI) seems quite stable, already.

The main point of discussion with the template, however, refers to the observation, that there was a large overlap between scenarios and functional requirements. Usually in phase 2 from each scenario only one functional requirement was derived, which is not what we had expected in the beginning. Looking back, however, we feel that this is a quite natural and predictable effect, as the clustering of the overall system into features

zooms down the system on a level that is too detailed for using scenarios in the “standard” way.

Since it still seems appropriate to divide the process into two phases, we favour to stick to the scenario/functional requirements scheme in future projects. However, we will be less strict and ambitious with respect to the mutual differences between scenarios and requirements both in form and in content. Again, the problem of potential overlap between scenarios and corresponding functional requirements occur first of all in mature features.

Conclusions

In this paper we discussed an approach how to successfully assign additional staff in order to deliver high quality requirements specifications. We do not claim that this is the general approach, of course. But given certain restrictions and an appropriate project context, this approach is reasonable and promising. As important success factors we regard:

- A system similar to the one to be specified has already been developed before.
- Most of the information needed is already available. (Nevertheless, it is not obvious, where to find it.)
- Domain experts are available on demand, if only short-time.
- Foster regular and informal feedback, interaction, discussion, communication, and continuous improvement.
- Ensure permanent quality assurance.
- Convince domain experts and get their commitment.
- Provide RE experts’ advice both systematically and on demand.
- The described two phased approach helps to shift the specification task from domain experts to full time requirements analysts.

At first glance, the overall effort of about 100 person months to obtain the requirements specifications needed may seem quite high. Nevertheless, we are convinced that this expense represents an excellent investment. The point is that the obtained high quality, uniform, and well structured requirements stored in a single requirements management database can be efficiently reused in subsequent development cycles for future passenger car models. This partly reuse of specifications promises two major advantages: better specifications in less time.

Future work will focus on the following aspects:

- Further improvement of the methodology described in this paper, as there are lots of other business units in need to take this important first step.
- Getting a better understanding what makes requirements reusable and how to do so.
- Support the reuse of requirements.

References

- [1] B. W. Boehm: “Software-Engineering Economics”, Englewood Cliffs, Prentice Hall, 1981.

- [2] J. Eisenmann et al.: "Entwurf und Implementierung von Fahrzeugsteuerungsfunktionen auf Basis der TITUS Client Sever Architektur. VDI Berichte (1374) 1997.
- [3] G. Kotonya and I. Sommerville: "Requirements Engineering", Chichester, John Wiley & Sons, 1998.
- [4] B. L. Kovitz: "Practical Software Requirements", Greenwich, Manning, 1999.
- [5] S. & J. Robertson: "Mastering the Requirements Process", Reading, Addison-Wesley, 1999.
- [6] A. Miller: "Managing Outsourced Embedded Systems Development" in: *Proceedings of Embedded Systems Conference*, San Francisco, Spring 1998.
- [7] K. E. Wiegers, "Software Requirements", Redmont, Microsoft Press, 1999.
- [8] T. Zink: "Konzept und prototypische Realisierung eines Werkzeugs zur Erstellung von Anforderungsspezifikationen für Softwarefunktionen im Automobilbau", diploma thesis in German, University of Stuttgart, 2001.

Acknowledgements

Our colleague THOMAS BEIL contributed an essential part to the overall success of our specification activities at MTC, especially in the important early project phase.

Most of the figures given in this paper were collected by THOMAS ZINK in his excellent diploma thesis [8] that evaluates our process.

About the authors

FRIEDEMANN KIEDAISCH (friedemann.kiedaisch@daimlerchrysler.com) is an industrial PhD student at the University of Ulm. He is working in the area of Requirements Engineering at the DaimlerChrysler Research Center in Ulm.

Company Description

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

relevant results to the business units of DaimlerChrysler.

Session 11 - SPI and Systems Design

**Session Chair:
Bernd Hindel, Methodpark,
Germany**

An Assessment Approach for User-Centred Design Processes	11-2
Assessing effects of RUP implementation using SPICE	11-15
Component-based development and distribution of a CAM product	11-26

An Assessment Approach for User-Centred Design Processes

Timo Jokela
University of Oulu

Abstract

We carried out a series of experimental assessments to learn how to perform effective assessment of user-centred design (UCD) processes. Our starting point was ISO 15504-based process assessment and ISO/TR 18529 UCD process model. During the experiments, the assessment style evolved towards a novel assessment approach. Specific characteristics of the assessment approach are a revised UCD process model, three process performance dimensions, and the implementation of the assessment as a workshop. The approach was found to be efficient and to give concrete, understandable and valid knowledge about the status of UCD. It also proved to be an effective instrument for training and increasing awareness of UCD.

Introduction

Usability is recognised as one of the most important quality characteristics of software intensive systems and products. Usability gives many benefits that can include "increased productivity, enhanced quality of work, improved user satisfaction, reductions in support and training costs and improved user satisfaction" [1].

The prevailing paradigm of developing *usable* products and systems is *user-centred design*¹, *UCD*: usable products are created through processes of user-centred design. There exist literature on UCD, e.g. books such as [2], [3], [4], [5], and the standard ISO 13407 [1]. All of them incorporate the same basic principles for developing usable

¹ Called also human-centred design [1], usability engineering [2]

products and system, such as user involvement, iterative design and multi-disciplinary teamwork.

UCD processes, however, are not typically largely present in software development projects. Or if there are some, their strategic impact in development projects is often invisible. Many products that are today in the market - or systems that are in use - represent poor level of usability. The challenge to improve the position of UCD in development organisations has been recognised in many presentations and panels in HCI conferences and seminars. There have been papers [6], tutorials [7], panels [8] and interviews [9] at CHI conferences. A European TRUMP project has also recently addressed this topic [10].

In our research, we use the recent standard *ISO 13407* as the main reference of UCD. It is recently published, and is not bound to any specific techniques or methods. ISO 13407 represents the prevailing and generally accepted understanding in the community of usability engineers and scientists of how usable product are created. The standard identifies four main processes of UCD illustrated in Figure 1.

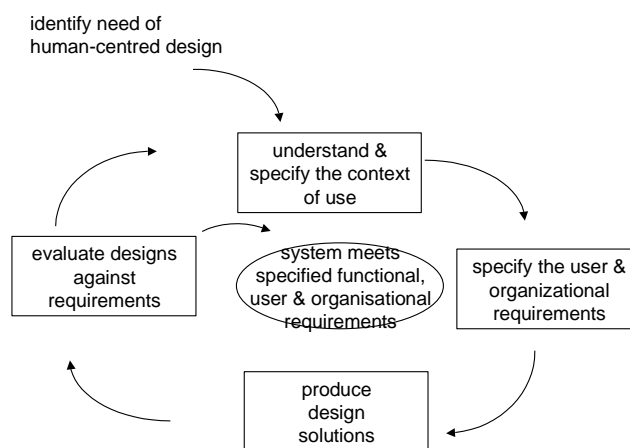


Figure 1. Processes of user-centred design, UCD (ISO 13407)

The processes can be briefly described as follows:

- *Understand and specify context of use.* Know the user, the environment of use, and what are the tasks that he or she uses the product for.
- *Specify the user and organisational requirements.* Determine the success criteria of usability for the product in terms of user tasks, e.g. how quickly a typical user should be able to complete a task with the product. Determine the design guidelines and constraints.
- *Produce design solutions.* Incorporate HCI knowledge (of visual design, interaction design, usability) into design solutions.
- *Evaluate designs against requirements.* The designs are evaluated against user tasks.

Research problem

A typical approach to start organisational improvement effort in any domain is to carry out *current state analysis*. Through current state analysis, one can identify the strengths and weaknesses of an organisation, and thus get a good basis for planning and implementing improvement actions. When we examine the recognised *process improvement models* of software engineering, for example IDEAL of Software Engineering Institute [11] and ISO/TR 15504-7 [12], they essentially include 'current state analysis' as a step in an improvement process.

In software engineering, current state analysis is a widely used practice in the form of *process assessment*. Well-known approaches for software process assessment are *CMM* [13], *Bootstrap* [14], and *ISO 15504* [15], formerly known as *SPICE*.

A number of different approaches have been proposed for the current state analysis of UCD, or as we call, *usability capability assessment (UCA)*, too [16]. The first *UCA approaches* were introduced in the first half of 90's. A remarkable achievement in this area was the publication of ISO/TR 18529 [17] in 2000 which defines the UCD processes in the format that complies with the requirements of ISO 15504. The UCD substance of the ISO/TR 18529 model is mainly from ISO 13407.

In spite of the many approaches, usability capability assessments are not yet a very widely performed activity. According to a study of Rosenbaum & al [6], 16 organisations out of 134 (12%) reported using 'organisational audits' as a means for enhancing 'strategic usability'. In the same study, audits were ranked to be one of the least effective approaches in promoting strategic usability.

Our research problem is to learn how to effectively perform usability capability assessment in industrial settings.

Motivation

Our research background is usability capability assessments that were carried out using the ISO 15504-style software process assessment at Nokia Mobile Phones, Teamware and Buscom. In these assessments, we used ISO/TR 18529 – or its predecessor *UMM Processes* [18] - as the UCD reference process model.

We gathered feedback from the assessments from the organisation and assessment team. The feedback indicated that there is space for improvement in the assessment approach. The following kind of problems were identified:

- The staff perceived the concepts of ISO 15504-style assessment difficult to understand
- The results were perceived not concrete enough
- There were disagreements among the assessors on the interpretation of the processes (especially on some base practices)
- There were disagreements about the ratings of the base practices
- The assessment was found very laboursome and even frustrating by the

assessment team

- Those who were not interviewed were in a ‘worse’ position than those interviewed (did not learn, did not get committed)
- A lot of discussion was about other than the substance of UCD (i.e. managerial issues)

Development of An Assessment Approach

The development of an enhanced assessment model took place stepwise in the three assessments at Nokia Networks and Nokia Mobile Phones. In this section, we describe briefly the assessments. The assessment approach that evolved is described in the next section.

First Assessment

Based on the earlier assessment experiences, the main research driver of this assessment was to develop more unambiguous interpretations of (the base practices of) the UCD processes.

The interpretation of the base practices realised to be quite a challenge. Jointly with the usability experts of the customer, we decided to carry out the assessment based on concrete *outcomes* of processes. For that, we redefined the outcomes of the ISO/TR 18529 processes to include only concrete deliverables.

The outcome driven assessment led also to a new kind of *performance dimensions*: *quantity*, *quality*, and *integration*. In addition, the different process model and performance dimensions meant a different way of presenting the results. We present the performance profile in one visual picture, using different symbols to denote the different dimensions.

We call these models as *KESU UDC Process Model* and *KESU Process Performance Dimensions Model* (KESU is the name of our research project).

A clear feeling after the assessment was that “we want to try this approach again”. However, there were also places to improve. The overall role and position of an assessment in the organisational context should be rethought. One organisational problem was, for example, that very few developers attended the opening session of the assessment. In addition, we found that definitions for the levels of rating the performance dimensions should be created, and some terms should be made easier to understand.

Second Assessment

This assessment had a different object domain than the earlier ones: the object of the assessment was the development process of customer documentation. The assessment had one significant new feature: it was decided to carry it out in a half-day workshop. The reason for this was practical: there was no time for an assessment with multiple interviews.

We used the process model and performance dimensions developed in the previous assessment. In the interview part, we asked the customer to describe the processes of the development process of the user manual and quick reference guide. We did not describe our reference model beforehand but used it as our reference in the interview.

In the results session, the findings were contrasted against the process model. The lead assessor explained the model step by step, and interpreted the information received in the interview session against the model. We agreed on the areas where the organisation had strengths and weaknesses. The session was finished in about 1,5 hours. Some refinements and clarifications were made on the process model during the discussion.

One immediate feedback came actually when we finished the session. One of the participants said that she already had started to challenge herself how to implement improvements on one specific area. Other participants reported: "The assessment pointed out many issues to consider in the following documentation projects" and "Now we know that task analysis is important. We need to work on usability requirements".

The assessment team found the assessment as a positive experience. We succeeded in getting a credible picture of the development process in a short time. We had felt that there was a positive atmosphere in the workshop.

The specific implication of this assessment to us was to try the workshop approach again. We found it efficient – one workshop instead of a series of interviews. Moreover, a workshop may be a solution to one problem that we have faced: people being in different positions in assessments (those who are interviewed, and those who are not).

Third Assessment

Before the third assessment, we identified different organisational contexts where improvements in UCD may take place, Figure 2:

- *Improvement of UCD at strategic level* means the creation of *UCD strategy*, and keeping it up-to-date. UCD strategy may include issues such as vision of the organisation in terms of UCD (e.g. compared with competitors), definitions of UCD policy, and selection of pilot projects where concrete improvement actions could be carried out.
- A typical way of introducing new methods into organisation is to experiment them in *pilot projects*. The project manager and other key persons of the project should be voluntary to try user-centred methods. As a result of a pilot project, the organisation has experience and knowledge about the usefulness of methods.
- *Improvement in the UCD infrastructure* could be, for example, development of company style guide, documentation of UCD methods and guidelines in quality manuals, and development of UCD training material for personnel.

We find that the different improvement contexts mean important implications to assessment. The improvement process in each improvement context may include an assessment activity. The focus of assessment, however, is different in these different

contexts. At strategic level, the assessment should support the creation of UCD strategy in the organisation. In pilot projects, the assessment should support planning UCD processes and use of UCD methods in a pilot development project. At the level of UCD infrastructure, the assessment should give a basis for setting up appropriate infrastructure improvement actions.

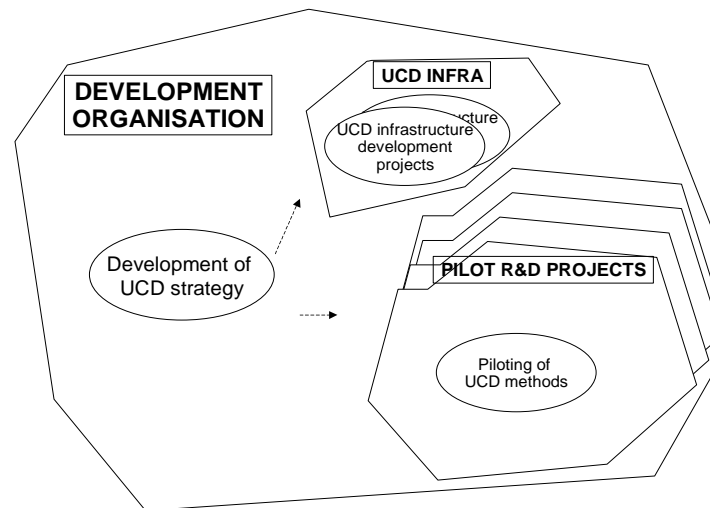


Figure 2. KESSU improvement model: UCD improvement actions may take place in different contexts in organisations

It was easy to find the appropriate improvement context for the third assessment: it was clearly a pilot project.

In the third assessment, we analysed the status of UCD of a previous representative project – that we call *baseline project*. As in the previous assessment, we first had an interview session. After that the assessment team had a wrap-up session that was experienced to be fluent: the team felt that it was easy to agree on findings. In the results session, the findings were contrasted with the process model.

As a basis of the results of the assessment, the project team planned together with the assessment team the UCD activities and methods that are reasonable to implement in the pilot project, Figure 3. This paper is written after the planning phase – the implementation of the plan is just about to start.

The assessment team felt that they had learnt about the UCD in the project as much (or even more) as we did in the earlier assessments with numerous interviews. In the end, we found as a very good sign that planning activities for the pilot project truly started.

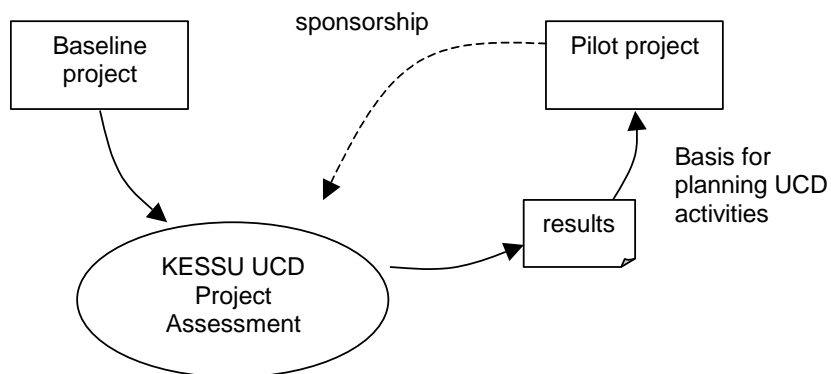


Figure 3. A past (baseline) project is assessed to give a basis for planning UCD processes in a customer project.

The Assessment Approach

In summary, our assessment approach for UCD processes evolved to have the following main characteristics:

- A refined UCD process model, *KESSU UCD Process Model*
- Three performance dimensions, *KESSU Process Performance Dimensions*
- *Visual representation of results*
- An assessment method where the assessment is implemented as a workshop, *KESSU UCD Project Assessment*
- Always including all UCD processes (that we currently identify six ones) in the assessment

UCD Process Model

The process model is illustrated in Figure 4. We identify six main processes: Identification of user groups process, Context of use process (multiple instances), User requirements process, User tasks design process, Produce user interaction solutions process, and Usability evaluation process.

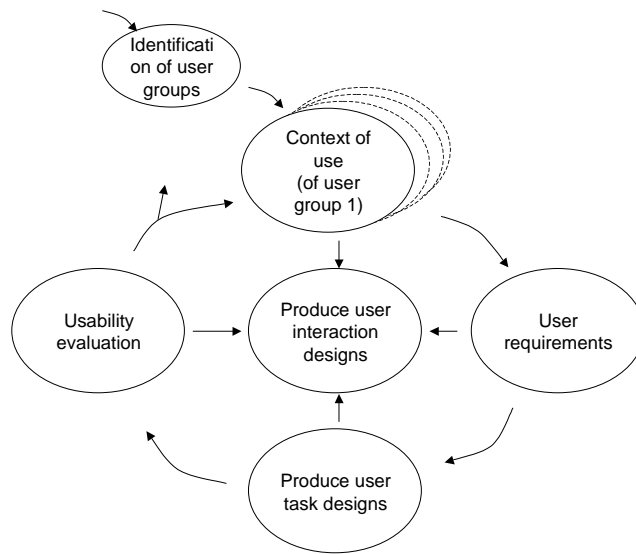


Figure 4. The KESSU Process model (ref. Figure 1)

The difference between the definitions of processes between the KESSU model and the ISO/TR 18529 model is illustrated in Table 1 (example: Context of use process). When we contrast the process definitions, we can find that the outcomes of the KESSU model include only concrete deliverables.

Context of use process	
KESSU UCD process model	ISO/TR 18529
Identification of user groups	Define the scope of the context of use for the product system
Description of the characteristics of users	Analyse the tasks and worksystem
Description of the environment of use	Describe the characteristics of the users
Identification of user accomplishments	Describe the cultural environment/organisational/management regime
Description of user tasks	Describe the characteristics of any equipment external to the product system and the working environment
Identification of user task attributes	Describe the location, workplace equipment and ambient conditions
	Analyse the implications of the context of use
	Present these issues to project stakeholders for use in the development or operation of the product system

Table 1. Illustration of differences between outcomes and base practices. Example: Context of use process

Process Performance Dimensions

We identify three performance dimensions: *quantity*, *quality*, and *integration* as illustrated in Figure 5:

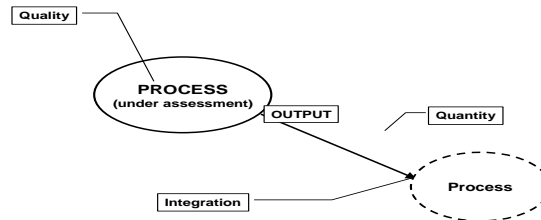


Figure 5. KESSU dimensions of process performance

- The *quantity* of outcomes of the process. The more extensively an outcome exists, the higher performance score it gets.
- The *quality* of the outcomes. With this dimension, we examine the quality and validity of the outcomes. For example, we want to make a difference whether an outcome is based on someone’s opinions or derived by using recognised user centred methods and techniques.
- The *integration* of outcomes with other processes. The more extensively the outcomes are communicated and incorporated in other relevant processes, the higher rating is given to integration.

Representation of results

In addition, the different process and performance models meant a different way of presenting the results. We use different symbols to denote the different dimensions, Figure 6.

	N none	P partially	L largely	F fully
Quantity				
Integration				
Quality				

Figure 6. Visual representation of performance dimensions

We present the performance profile in one visual picture. We use the process diagram of user-centred design as a background picture as illustrated in Figure 7.

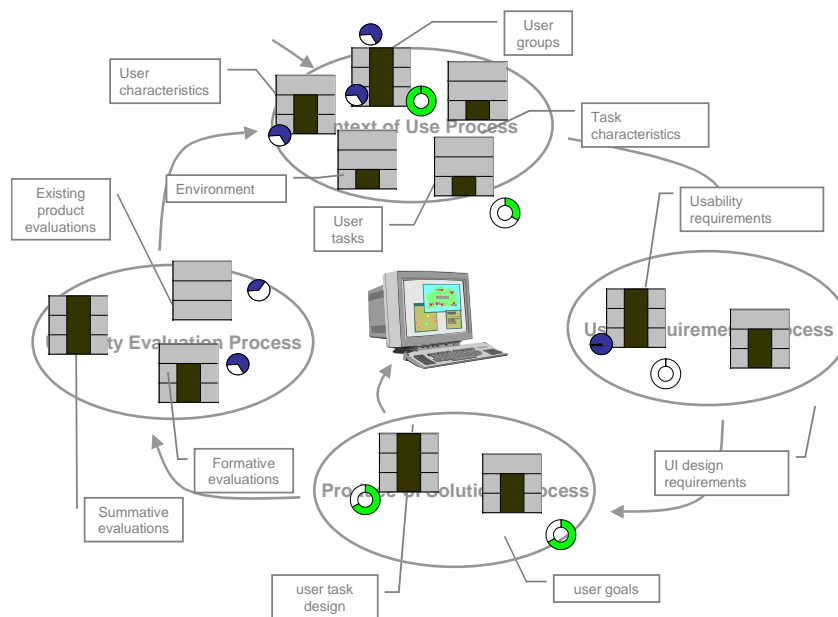


Figure 7. Example of performance profile presentation

Assessment method

An assessment is carried out as a workshop where all the persons of the pilot project should attend. The customer of the assessment is the project manager (or equivalent). A past project (or a project that is close to end) is selected as the baseline project for the assessment. The previous project should be such that is found as a relevant and representative project for the key persons of the customer project. In an ideal case, the baseline project is carried out with the same management and staff as the new one.

Conclusions and Discussion

Our target was to learn how to perform assessments of UCD processes through experiments in industrial settings. In the beginning of the research, we had experience on traditional process assessments. In the end of the research, we identified three different assessment types, and developed an enhanced assessment approach to one type: UCD project assessment. It has an outcome-driven process model, three performance dimensions, and an implementation the assessment as a workshop. We always included all the UCD processes in the assessment.

The approach means some differences contrasted with traditional process assessment. We focus on the (UCD) substance – not on the management - of processes. We examine the not only the extent of practices but also the quality how they are carried out, and how the results are integrated into design decisions.

Our experience on UCD project assessment shows that the new process and performance models seem to make the assessment and UCD more understandable to the audience and

easier for the assessors in our context. The workshop type of assessment makes an assessment efficient and spreads UCD knowledge to larger part of the staff than interviews of a limited number of people.

We want to emphasise that our assessment approach is a complementary one to the traditional process assessment. Compared with ISO 15504 assessment, we can say that we examine the UCD processes thoroughly 'below the level 1 of capability'. Traditional process assessment should be used in contexts where it is applicable to examine the management of UCD processes at higher levels of capability.

Limitations

Our goal for the assessments was to give a good basis for improvement actions in UCD. From this viewpoint, issues such as spreading knowledge about UCD to the staff and getting them committed to UCD improvement actions are important. A different target for assessment could be to get exact ratings of usability capability for selection of contractors. This was not our goal. Another viewpoint that we excluded is standardisation that has been one driver of the ISO/TR 18529 and HSL models.

Another limitation is related to the assessment with the ISO/TR 18529 model. The assessments were very much based on the interpretations, experience and style of the lead assessors. The interpretation of the ISO/TR 18529 model is based on documentation. Some other person may have conducted the assessments in a different way. A specific feature in our assessments is that most organisations represented geographically limited industrial settings (Oulu region in Finland).

Implications for Next Assessments

We will continue with the KESSU process model and performance scale (probably with refinements) in our next assessments. We find that the outcome-driven process model makes discussions concrete; the three-dimensional performance scale identifies different kind of problems in the effectiveness of UCD and makes possible to have an appropriate level of abstraction in the assessment, and implementation as a workshop spread knowledge of UCD to the staff and gets everybody involved.

We find that each new assessment should be considered a research activity. A researcher should try to get access for following assessments and for gathering feedback from them. There is definitely space for improvements both at the model and in the assessment method (steps of assessment) levels. We will carry out further assessments, and regard each of them also as a research effort.

New Research Topics

The next step in our research is of very constructive nature: to document the assessment approach as a handbook. We assume that the creation of such a document is not a one-time effort but it will be revised after trials. Another artefact we plan to develop is a template to make the assessment efficient. We hope to have an online documentation in the workshops, and deliver the results immediately.

One interesting challenge is how to reliability verify the success of an assessment. We have gathered a lot of feedback in our assessments. However, it still is difficult to make definite conclusions. For example, we find that the assessments should be also training occasions where the understandability of models and results is important. Some others may disagree with the importance of this criterion.

Acknowledgements

There are a number of people who have contributed this work. I wish thank the project members of the KESSU project: Netta Iivari, Mikko Jämsä, Tonja Molin-Juustila, Tero Posio, Mikko Rajanen, Samuli Saukkonen, and Kari Kuutti. I also wish to express thanks to the personnel of the partner companies Buscom, Teamware, Nokia Networks, and Nokia Mobile Phones; and to the national funding organisation TEKES.

References

1. ISO13407, *Human-centred Design Processes for Interactive Systems*. 1999, International Organization for Standardization, Genève, Switzerland.
2. Nielsen, J., *Usability Engineering*. 1993: Academic Press, Inc. 358.
3. Hix and Hartson, *Developing User Interfaces : Ensuring Usability Through Product & Process*. 199.
4. Holtzblatt, K. and H. Beyer, *Contextual Design: Defining Customer-Centered Systems*. 1998, San Francisco: Morgan Kaufmann Publishers. 472.
5. Mayhew, D.J., *The Usability Engineering Lifecycle*. 1999.
6. Rosenbaum, S., J.A. Rohn, and J. Humburg. *A Toolkit for Strategic Usability: Results from Workshops, Panels, and Surveys*. in *CHI 2000*. 2000. The Hague: ACM.
7. Bloomer, S. and S. Wolf. *Successful Strategies for Selling Usability into Organizations*. in *Companion Proceedings of CHI '99: Human Factors in Computing Systems*. 1999. Pittsburgh, USA: ACM Press.
8. Rosenbaum, S. *What Makes Strategic Usability Fail? Lessons Learned from the Field*. in *CHI '99*. 1999. Pittsburgh, USA.
9. Anderson, R., *Organisational Limits to HCI. Conversations with Don Norman and Janice Rohn*. *Interactions*, 2000. 7(2): p. 36-60.
10. Bevan, N. and J. Earthy. *Usability process improvement and maturity assessment*. in *(in these Proceedings)*. 2001.

11. McFeeley, B., *IDEAL SM: A User's Guide for Software Process Improvement*. 1996, Software Engineering Institute: Pittsburgh. p. 236.
12. ISO/TR15504-7, *Software Process Assessment - Part 7: Guide for use in process improvement*. 1998, International Organization for Standardization, Genève, Switzerland.
13. Paulk, M.C. and al, *The Capability Maturity Model: Guidelines for Improving the Software Process*. 1995: Addison-Wesley.
14. Kuvaja, P., et al., *Software Process Assessment and Improvement - The BOOTSTRAP Approach*. 1994, Cambridge, MA: Blackwell Publishers.
15. ISO/TR15504-2, *Software Process Assessment - Part 2: A reference model for processes and process capability*. 1998, International Organization for Standardization, Genève, Switzerland.
16. Jokela, T. *Usability Capability Models - Review and Analysis*. in *HCI 2000*. 2000. Sunderland, UK.
17. ISO/TR18529, *Human-centred Lifecycle Process Descriptions*. 2000, International Organization for Standardization: Genève, Switzerland.
18. Earthy, J., *Usability Maturity Model: Processes*. 1999, Lloyd's Register of Shipping.

Assessing effects of RUP implementation using SPICE

Göran Grahn

Håkan Wickberg

Volvo Information Technology, Gothenburg, Sweden

Introduction

Background

Volvo Information Technology (Volvo IT) is the Volvo Group's resource and expertise centre for IT systems. We are also an active and dominant supplier to Volvo Cars, owned by Ford Motor Company since April 1999.

The Methods & Techniques department is responsible for supporting application development and maintenance teams with development processes, methods, tools and application development environments. In 1999 we started to introduce a set of new processes, methods and tools.

The strategy we have for introducing new methodology is to aim for:

- Establish a common process for Application Development within Volvo IT
- Integrate Business Engineering and Application Development
- Replace the old development methodology with state-of-the-art methods and tools supported globally

The two main parts of the new methodology is the Project Control Model (PCM) developed in-house, and the Rational Unified Process (RUP) from Rational Software.

The expected effects of introducing PCM and RUP are:

- More projects on-time and within budget
- Shorter lead time until delivery of first increment
- Better product fit to actual needs at time of delivery
- Reduced rework costs
- Better product maintainability

The Challenge

The total number of developers to be trained in RUP is estimated at 800 people. This represents an investment in the range of 50 – 100 million SEK, equal to 5 – 10 million Euro. So the question from top management was no surprise: "Can you prove to us that RUP is having effect?"

Well, what could we measure?

- Just looking at delivery precision (Time and Cost) of projects using RUP would not be enough to judge the effects of RUP
- The questionnaire we designed for providing feedback on RUP pilots would apparently not give an objective answer for – It's quite normal to give positive reaction when you are among the first to work with a new process and using new tools
- Then we came across the Software Process Improvement and Capability Determination (SPICE) Framework for assessing "process capability". By assessing the "traditional way of working" of a project team, and comparing with an assessment of "working with RUP", we would hopefully be able to document proof of process improvements.

So, let's have a brief look at RUP and SPICE as a basis for discussing the assessment programme and the experience we made.

RUP – The Rational Unified Process

RUP overview

The Rational Unified Process [1] is "a controlled iterative process for application development", which is based on the following "six best practices":

- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually, using the Unified Modelling Language (UML)
- Continuously Verify Quality
- Control Change

The iterative approach helps you deal with risks throughout the project

- Architectural risks are dealt with early in the project, and errors are corrected over several iterations
- Integration is not one “big bang” at the end of the project, instead elements are integrated progressively.

See end of article for a reference to more information about RUP.

How we implement RUP at Volvo IT

The experience from six small pilot projects using RUP in 1999 made us realise that introducing a new development process is a large investment in building competence. Following the recommendation from the Methods department, Top Management decided on a “project by project” implementation strategy, starting with 10 quite small projects in 2000, and continuing with some 25 projects in 2001. To be able to support the projects, we started training people to become “RUP coaches” and “RUP specialists”. At the beginning we filled the specialist role with external consultants, but as experience grows, they are being replaced with in-house staff.

Key to our success has been the program of “Workshops and Reviews” we offer each project. We all know that what was clear at the training course often is difficult to apply to your own project two or three months later. The purpose of the workshops is for each new major activity to refresh the theory, and to help the project team in applying the theory on their own problem. In addition to the workshops and other coaching activities, our coaches and specialists also perform a set of reviews, aiming at assuring that the project builds the right product, and that the process and methodology is understood and applied in the right way.

The SPICE Framework

SPICE overview

SPICE (ISO/IEC TR 15504) [2] provides a two-dimensional model of processes and process capability that forms the basis for process assessments. Processes are grouped into five categories:

- Customer – Supplier processes, e. g. Requirements Elicitation
- Engineering processes, e. g. Software Construction
- Supporting processes, e. g. Configuration Management
- Management processes, e. g. Project Management
- Organisation processes, e. g. the Process Assessment process itself

Within each category, processes are described by a statement of the purpose of the process, which includes an outline of the intended outcomes of process implementation.

For each process, the process capability is assessed using nine attributes, examining the performance of the process, how it is managed, what work products are produced and how changes are managed etc. The rating of these attributes is used to derive the capability level for the process. Each process receives a separate capability level rating.

SPICE has a lot in common with the Capability Maturity Model - CMM (TM) [3] from the Software Engineering Institute (SEI) of Carnegie Mellon University, US. The main difference is that SPICE allows you to select which process you want to assess, and each process is rated by itself, where as CMM has a "packaging" of processes that must be performed according to certain requirements in order for the Organisation to be at a certain maturity level. The new CMM Integrated (CMMI) offers a similar approach as SPICE.

See end of article for references to more information about SPICE and CMM.

Establishing Target Capability Profiles for RUP

So, it should be possible to establish a "target capability profile" that an organisation would achieve if it followed RUP in an idealistic way. As a matter of fact such a target profile existed within Rational Software [4], as a result from an internal study in 1998 by John Smith of Rational Software. This study assessed an older version of RUP using an older version of SPICE.

In the spring of 2000 a new "target capability profile" for RUP version 5.5 was determined using the current version of SPICE at the University of Borås, Sweden [5]. The result of this study shows that capability level 3 is expected for a set of selected processes in an idealistic situation when following the RUP approach. However, this is expected to be achieved on a 2 – 3 year time horizon for a team working with RUP. We can not expect project teams to reach level 3 on their first project using RUP if they set out at level 1 or 2.

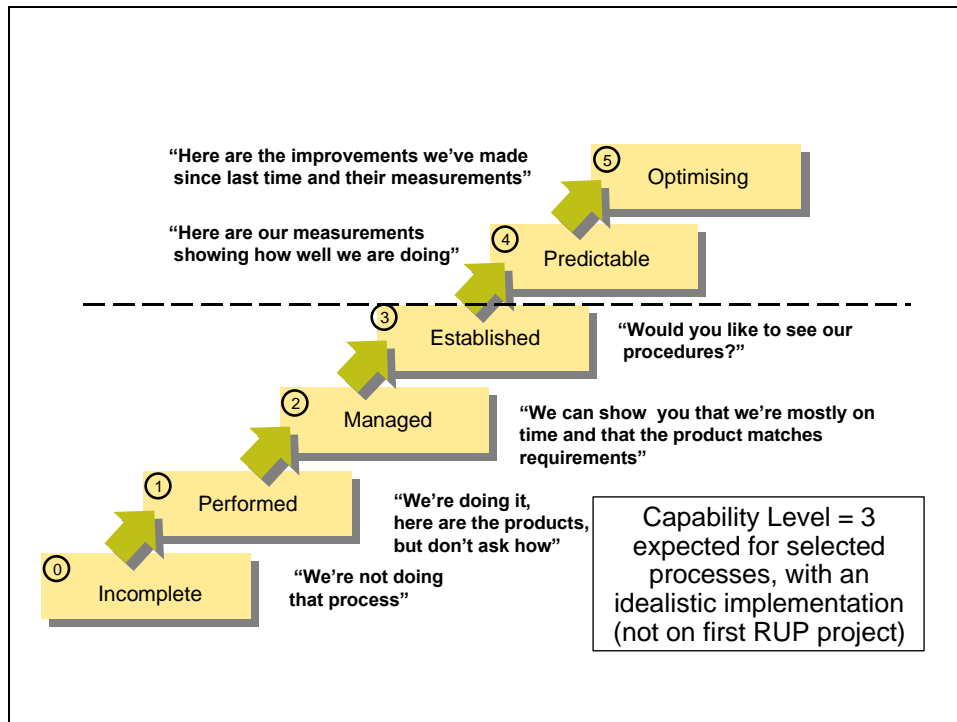


Fig. GGRAHN.1: SPICE – Capability Levels “in practice”

Assessment of RUP Implementation

Assessment Approach

Volvo IT set up an assessment programme to determine the “Before / After” capability profiles for three of the ten RUP project teams. We had two purposes:

- To provide input to decision on continued implementation of RUP at Volvo IT
- To get experience from using SPICE, to guide future assessments of software process improvement projects

The three projects were chosen to represent different application areas, different locations and different technical environments.

We engaged an external consultant to train us, and to lead the assessment team, and we planned our assessment programme with the following phases, according to the requirements of SPICE:

- Pre-Assessment Questionnaires, with collection of key documents
- Assessment plan and schedule
- 1-hour briefing of project team (one week before interviews)
- 1-day interviews of project team
- Validation and Rating
- Feedback of findings to project team (day after interviews)
- Assessment Report (one week after interviews)

The scope of the assessments in terms of processes assessed is shown in Figure 2 below. For all these processes, the expected capability level = 3 when using RUP 5.5 in an idealistic way.

For each project assessed:

- The “Before-assessment” or “the traditional way of working” was performed at the beginning of the project, looking at how the project is likely to have been performed had it not been selected to be one of our RUP projects,
- The “After-assessment”, assessing the way of working with RUP, was performed in the Construction Phase of RUP.

The interviews were limited to fit within one day, for reasons of disturbing the project team as little as possible, while at the same time gathering enough information to be able to give a correct rating. In one case one day was too short to assess all the processes we had planned to assess. We also discovered that in some cases it was too early to assess the Integration and Testing processes, since they had not been performed in the first iterations to the extent that a fair rating could be given.

Assessment Results

Below the assessment results for one of the projects is shown (Fig 2). For each process, the upper bar is “traditional way of working” and the lower bar is “working with RUP”.

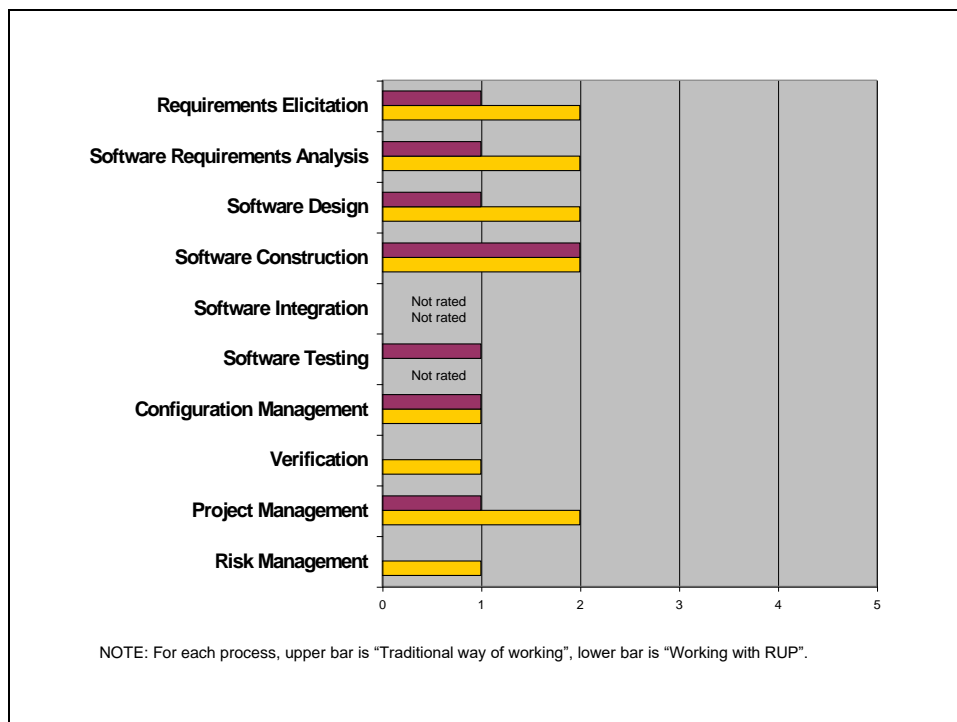


Fig. GGRAHN.2: Capability Levels “Before / After” for one RUP project

The diagram shows that the capability increased from level 1 to level 2 in the

Requirements, Analysis, Design and Project Management processes. The Verification and Risk Management processes were not performed at all in the traditional way of working. “Working with RUP”, they were rated at level 1.

In all of the three projects we assessed, the Requirements, Analysis, Design, Construction and Project Management processes were all rated at level 2 in “working with RUP” (with one exception). These are the processes that have been focussed on in the coaching of these first RUP projects. It is clear that the focus when starting to use RUP is on getting the end product right first, meaning that the engineering processes, modelling techniques and new tools are given most attention. In two of these projects the Windows DNA environment was used for the first time, adding more needs for training and coaching in technical issues.

The assessment result clearly indicates that the implementation of RUP is having effect. However, there is a big potential for further improving process capability with growing experience in RUP, and by enhancing the support given to RUP projects on the Management and Supporting processes. Suggested actions concerning both the content of the support as well as how the support is delivered are listed below.

Likewise, process improvement actions have been suggested to the project teams at the feedback sessions where the results of each assessment have been presented and discussed. Examples of suggested actions include enhancing test documentation, using risk lists, planning for handing over project results to maintenance etc.

The assessments offered a good opportunity for dialogue between the methods department and development projects, a chance to see what parts of the theory that works or not in real projects and a chance to clear out misunderstandings and learn what else would be needed to make the theory work.

Experiences gained

Feedback to continued implementation of RUP

Besides providing proof that the implementation of RUP is having effect, the assessments also gave valuable insights in strengths and weaknesses in how we are running the implementation. Some examples of actions that were suggested to strengthen the support we give to new projects starting to use RUP during 2001 are:

1. Resolve inconsistencies between RUP and our quality system on document issue and control mechanisms – That will make it easier to achieve a good management of Work Products
2. Introduce the Software Quality Assurance Plan available in the Project Management Workflow of RUP 2000 in the Volvo IT Development Case template – That will help project managers to start planning issues related to Quality Assurance early in the project

3. Consider “Just-In-Time” self-training modules, which could be 10-20 slide Powerpoint presentations available on the Intranet on issues like Reviews, Document Control, Iteration Management, Interaction between RUP and our Project Control Model etc.

Experience from using SPICE

Experience from the assessments performed within the programme can be summarised as follows:

- For the purposes of why these assessments were performed, the 1-day interview schedule we used was “good enough” to roughly capture the project capability profile for the selected processes
- Assessment results with suggested actions was very well received by project teams at feedback sessions the day after the interviews

The conclusion of the positive reception we met in doing these assessments, is that SPICE assessments should be offered to projects and to maintenance teams in the line organisation, as a service to determine the capability profile for relevant processes and to identify risks and improvement areas.

The “Before / After” approach used to assess the effects of implementing RUP would work to assess the effects of any software process improvement project. Which processes to assess would depend on the scope of the improvement project.

One more challenge ...

When we reported our findings to top management they were quite pleased, but could not hold back the question: “Can you tell us exactly how much money we will be saving?”

Well, that’s a tough one to answer ... However, the April 2001 issue of the IT Metrics Strategies contains an article [6], discussing how much faster, cheaper and better a team working at CMM level 2 is compared to a team working at CMM level 1. The article is based on aligning statistics from CMM assessments to the large database run by Quantitative Software Management (QSM), containing productivity measurements from over 2500 projects. While this article discusses large projects, it also refers to data presented by QSM [7] which shows that moving from CMM level 1 to CMM level 2 when coding and testing a business application of some 50000 lines of code, will

- Reduce the schedule by 17 %
- Reduce the effort by 46 %
- Reduce defects by 51 %
- And, moving on from CMM level 2 to CMM level 3 will again reduce effort and defects by some 50 %!

Now, CMM maturity levels are not exactly the same as SPICE capability levels, but the message is clear: Software Process Improvements will have a very positive effect both on

Time and Cost, as well as the Quality of the products we deliver to our customers.

These statistics were accepted by top management as an indication that the results we are making in software process improvement will provide a good return on investment.

Conclusions

A short summary of what we are doing is:

- Volvo IT is investing a considerable amount of money in implementing RUP as the new common process for application development
- Top management wants to know if it is a good investment. – Is there a way of proving that RUP is giving the expected effects?
- We have used the SPICE Framework to demonstrate the improvements achieved by three of the project teams that we have coached in the use of RUP
- There are statistics available that translate an increase in process capability to savings in Time and Cost and to improved Quality

We now have a decision on establishing a team of SPICE assessors within Volvo IT that will offer assessments with different approaches. One approach will be to focus on one or a few processes at a time. Examples include Project Management and Risk Management or the Configuration Management process. With such a narrow scope, a number of projects or maintenance teams can be assessed at a limited cost.

A second approach will be broader assessments of large, business-critical and high-risk projects. For each of these projects the scope regarding what processes to assess will be decided, and the time and resources needed to perform the assessment will be planned.

A third approach would be to copy the assessment programme we used for assessing the effects of RUP, to assess the effects of other software process improvement initiatives. Which processes to assess would depend on the scope of the improvement.

We have found the SPICE Framework very useful to make the effects of software process improvement visible through highlighting both results achieved as well as remaining improvement areas.

References

- [1] A brief description of the Rational Unified Process (RUP) can be found in the article “Best Practices for Software Development Teams”, available at www.rational.com/products/whitepapers/100420.jsp
- [2] Information about the SPICE Framework is available on the official homepage of SPICE at www.sqi.gu.edu.au/spice

- [3] Paulk M. C. et al. The Capability Maturity Model – Guidelines for improving the Software Process, Addison-Wesley, 1995.
- [4] The assessment of RUP 5.0 against an older version of SPICE is available at www.rational.com/products/rup/resource_center/media/RUP15504final.pdf
- [5] Jakobsson Marie, Predicting software quality with ISO/IEC TR 15504 – Capability determination of the Rational Unified Process, Master Thesis of Informatics 2000:M17, Department of Computer Science and Business Administration, University of Borås, Sweden.
- [6] Rifkin Stan., Climbing the SEI CMM Makes a Difference on Software Projects, in: *IT Metrics Strategies, April 2001*, The Cutter Consortium
- [7] Putnam Lawrence H., Linking the QSM Productivity Index with the SEI Maturity Level, available at: www.qsm.com select Resources, look under White Papers

Component-based development and distribution of a CAM product

Massimo Capra, Michela Oggioni
Fidia S.P.A.
Corso Lombardia, 11
10099 San Mauro Torinese, Italy

1. Introduction

Component based development assumes that application development can be significantly improved if applications can be quickly assembled from pre-fabricated software components [1, 7]. Application development has the potential to move from a craft activity, as it is in most of the cases now, to an industrial process, fit to meet the needs of higher productivity, higher quality and lower costs.

However, several issues have to be resolved yet [3,2]: portability, interoperability, quality of components. The project described hereafter addresses a simpler case, because all components are developed internally. Yet, all of the above issues have to be addressed.

Our company develops two software products. In 1997 we decided to redesign them as a set of high level components, easy to plug in. The goal was to ease the customization process for customers; to ease the corrective maintenance of modules; to ease the adding of new functionalities (either by adding modules, or by adding functions to existing modules).

Each module provides either base functionalities not directly available to the user (e.g. an input interface converter or a mathematical library) or functionalities directly available to the user (e.g.: roughing toolpath computation or toolpath simulation and workpiece rendering).

Building and maintaining our software products is easier today than 2 years ago thanks to the component approach. However, the assembling process is still on our shoulders, and still time consuming.

While functions are added or enhanced on the demand of the market or of specific customers, each customer subscribing a license update contract receives at every release date the latest version of the whole product, including all functions. This eases the versioning process. On the other hand customers sometimes are forced to upgrade their version to receive functions they do not need and that make the product bigger.

Another major problem, according both to the customer's point of view and to our perspective, is user documentation. We have to upgrade user documentation at each upgrade, and distribute it to all customers. And customers have to upgrade, even if the enhancements regard functions they are not concerned about.

Observing the spread of e-commerce and e-business on the Internet, we notice that, under conditions of highly automated customization facilities, a customer can accept the burden of defining his own product. For instance, when buying a ticket for a flight, the customer asks its travel agent for different airlines, schedules, connections and prices: he/she has the only burden of knowing exactly where and when he/she needs to go, and how much money he/she wants to pay. The travel agent provides him with the ticket. With e-ticketing, all the task of arranging connections and prices is performed by the customer, supported by an effective web-based system.

Our goal now is to apply this approach to our products too. In our component based scenario the customer asks for a set of predefined CAM related functionalities to the component e-agent, that provides him with the components he/she needs. This approach increases system responsiveness and flexibility and decreases management costs.

The paper is organized as follows. Section 2 describes in more detail the relevant characteristics of the company and the products. Section 3 presents the overall vision of our component based approach to the development and distribution of components. Section 4 presents the infrastructure that supports development and distribution of components.

2. Company Context

The work we describe is being performed at FIDIA S.p.A., a company based in Northern Italy that produces numerical controls (NC), high speed milling machine tools and CAM (Computer Aided Manufacturing) systems. The company, together with its subsidiaries all around the world, employs 250 people; 40 of them deal with software development and maintenance.

Software is a strategic factor in this area:

- machine tools, whose costs are up to ten times the cost of their software, need highly reliable software
- software upgrades need to be produced in short times, since requests of customers are related to software most of the times
- since the lifecycle of software packages is in the order of years, maintenance is an

important cost factor.

Software development in FIDIA applies to three domains:

- Numerical Control: real-time embedded software that controls the machine dynamics and parameters;
- User Interface: monitoring software, with no real-time constraints, that allows the user to set and retrieve data and to start part programs execution;
- CAM: part program generation software. This software computes sequences of movements (part programs) that a tool must follow in order to mill a user defined part.

In order to have the part milled, the work flow starts from CAM software with the computation of the tool paths (collected in standard format files) on the mathematical model of the part, then, by means of the user interface module, the user notifies to the NC the data it needs to have the machine tool milling those trajectories (i.e. tools shape and dimensions, maximum speed of the axes movement, ...). According to the materials to be milled (cast iron, steel, marble, ...), that depend on the customer field of application, different tools and sequences of part programs have to be managed.

Our company develops two CAM software products: HI-MILL for complex surfaces toolpath generation, ISOGRAPH for 2.5D machining. The work described in this paper applies to the CAM products.

3. Overall Approach

Currently we have two products (HI-MILL and ISOGRAPH). The customer installs and uses one of them, or both, each one has a slightly different user interface. Each one is made up of subsystems (or sets of C++ classes that we call components). We have at present 16 HI-MILL components and 8 ISOGRAPH components. Each component ranges in size from 8000 to 14000 lines of code, from 15 to 24 C++ classes.

Each product evolves during time, usually we release three to four versions per year. However, customers can request and obtain special versions of the products. This increases the number of versions we have to deal with.

We are in the process of reengineering the products, with this approach:

- Define a common user interface for the two products. In a certain sense this means merging the products, however this is not completely true, as it will become clearer later.
- Transform the subsystems in CORBA components. Each component exports a set of functionalities, or commands.
- Make it possible for a customer to assemble the components he or she needs, thus obtaining a final product that is more suitable to his/her needs. For instance, a customer could assemble four components from the product now called HI-MILL, three from ISOGRAPH. Since some combinations are not meaningful, and some components are mandatory, we envisage that a total of 12-15 different tailored products could be built by assembling them in various combinations. All combinations share the same user interface, more or less rich in available

functionality.

- The customer can download the components from the web site of our company, and plug them together at his/her premises, without intervention from our company. An intelligent agent, function of the components downloaded and automatically generated, guides the composition process.
- With a similar procedure, the customer downloads and composes also the related user technical documentation.

The expected advantages from this approach are many.

- The customer deals with just one product, that contains exactly the functionalities he needs.
- We, the developers, have fewer versions of products to maintain. Actually, versioning and maintenance are at the level of components.
- When a new version of a component is released, the customer that use it should upgrade. But customers that do not use it are not obliged to upgrade. With the previous approach, a modification of a component meant a new version of the product, and therefore an upgrade for all customers. The same applies to user documentation.

The focal point of this approach is the repository of components (see Figure CAPRA.1), accessed internally by FIDIA employees, to develop and maintain them, and externally by customers via a web browser, to acquire and assemble them.

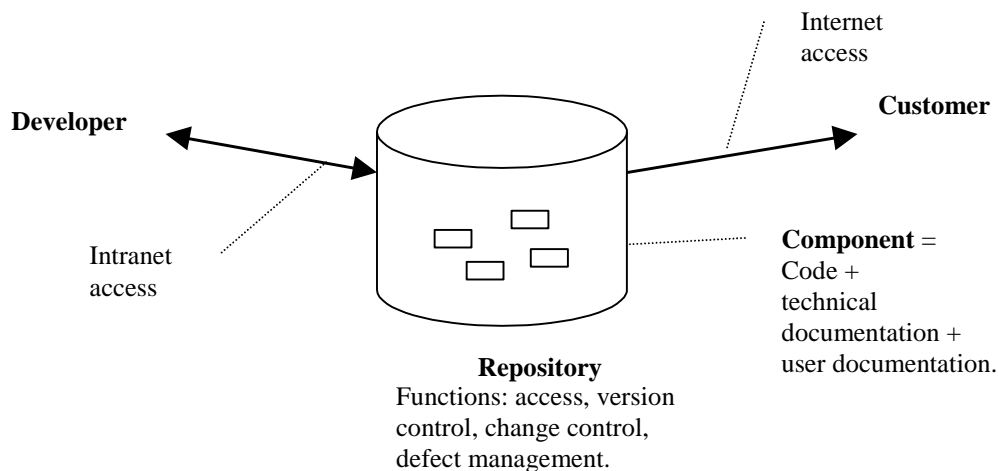


Figure CAPRA.1. The repository of components.

When a new customer acquires the CAM Programming Suite, he/she can choose the functionalities he/she needs via the web browser. He/she downloads the ‘functionalities’ and their user documentation in a component-based auto assembling application.

The status of each bought functionality (up to date, needs upgrade) can be monitored and component upgrades can be downloaded, together with the new documentation.

Also new functionalities (i.e. new components) can be downloaded, if the user needs them.

The intelligent agent that decides which components the user needs given the chosen functionalities and that assembles the components will not be addressed in this paper. The licence policy will not be described either.

4. The Component Infrastructure

We describe here the infrastructure we have built to support the development, distribution and maintenance of components. We will describe what components are, how they communicate and how they are packaged. Finally, we will describe the repository of components.

Component

A component is implemented through a set of C++ classes and publishes one or more interfaces via class definition. Each published class specification is CORBA compliant, obtained via IDL (Interface Definition Language) compilation.

It is possible to recognize a hierarchy of components (Figure CAPRA.2).

User interface. This component manages the interaction with the user, typically but not necessarily on a GUI, and requires services from other components. The user interface is a mandatory component. We refer to it as a particular component, since our need is to have all other components sharing this common interface. We chose this simpler approach, even if it is possible to consider the user interface as a component like all the others. User interface is a particular component, since its services can be called by any other component.

Librarian component. This component records the inter-component services exported by all the components of the system, and acts as the component glue (it connects the client of a service with the component exporting that service). It is initially identified as a part of the *User interface*. This component is mandatory, since it represents the glue of the application.

User component. This component satisfies requests from the user interface and from other components, if necessary. User components are visible to the end user, who has to explicitly require them when he/she acquires/installs the product.

Base component. This component satisfies requests from other components, but not directly from the user. It can not issue requests to non-base components.

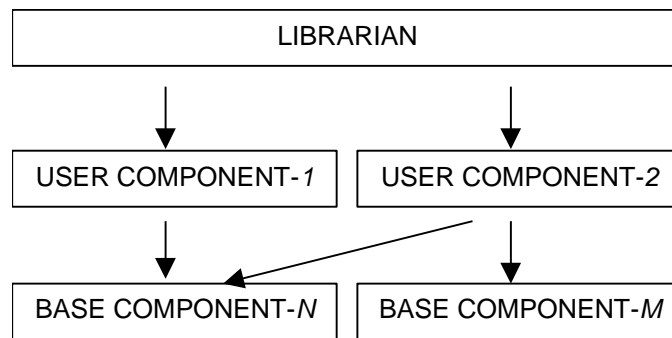


Figure CAPRA.2. The component hierarchy.

Publishing an IDL interface is necessary but not sufficient for a component to be pluggable in the system. It must also publish a *Command* Interface versus the User Interface component (see coming subsection on Communication with the User Interface), and register its services with the Librarian component.

Ideally, each component evolves independently from the others. From a syntactical point of view, the interface of each component must not change over time. From a semantic point of view, the function associated to an interface on a component should not change. This allows syntactical and semantic compatibility of each version of a component with any version of any other component. As a corollary, this allows also backward compatibility of a component with any previous version of it.

In practice, this is a hard constraint that we will strive to keep, but we will not be surprised too much if the evolution of the product will force us to break sometimes this constraint.

Further, during the evolution of the product some components could be merged and new ones could be introduced.

Component packaging

As already stated, each component publishes an interface, CORBA compliant. Communication between components is CORBA compliant and can take place only through this interface.

A component is packaged as a binary file. Actually, a binary file can contain more than one interface. There are two types of packaging:

- DLL: the binary file is a DLL exporting the component interface(s). This approach can improve performance.
- EXE: the binary file is an executable file responsible for registering its interface to the CORBA naming service. This approach allows components to be run in different address spaces (allowing for instance multi-CPU execution), but can penalize performance.

The first phase of the project deals almost only with DLL components, but we are evaluating for each component the opportunity of having it running as a separate

executable.

The only component that belongs to both types since the beginning is the Numerical Control component: it can be on a local or remote machine, being installed on the same host computer where the USER INTERFACE component is running, or on a computer connected to the host via a network. In order to improve the communication performance, in the local configuration the NC component is loaded from a DLL file (type A), while in the remote configuration it is an executable whose interface is retrieved via the CORBA naming service (type B).

Component loading

The *librarian* component, seen as a part of the USER INTERFACE module, is responsible for loading the components that compose the user application, as requested, and of the initialization of the communication.

In order to load the desired components, the librarian scans a configuration file (in ASCII format) containing the path name of each needed DLL or executable. For each line of the file, if the component is a DLL, it is loaded, if it is an executable, it is executed. Particular attention has to be paid to the automatic composition of the configuration file.

The loading phase is the only point in which it is necessary to distinguish the component packaging: if the component is of type B, the librarian launches it (if local) and retrieves its interface, represented as a (remote) object, via the CORBA naming service; type A components are loaded as DLL, and their interface, represented as a (local) object is retrieved directly.

After this initialization phase, a component communicates with any other via the same C++ syntax transparently on remote or local object representing inter components interface.

The librarian has then to establish the connection among components.

Communication between components

Components should communicate both with the user interface component and with other components. We will use two different communication styles for the two cases.

Communication with the User Interface

Each component will communicate with the user interface using the *Command* and *Factory* patterns [6].

A concrete Command is the specification of an abstract class declaring methods `run` and `undo` for a particular functionality of the component. A Command Factory is an abstract class mainly declaring methods for creation of a Command object. A concrete Command can be a part of a Composite Command pattern, that is a Command composed by more than one sub-command. Both abstract Command and Command Factory classes are derived from a CORBA IDL in order to allow component distribution.

A Command exports at least one method for asking parameters to the user via a standard GUI, if parameters are needed, one method for loading and one for storing its

parameters on file, for batch execution, one method for executing the command and one for undoing it.

The IDL of component interface declares the same function exporting the complete collection of Command Factories whose created commands will be available to the user. For the sake of simplicity we will not refer to Factories but directly to Commands in the following of this document. The use of the Factory pattern makes easy for instance the management of Command Lists, useful for implementing user defined macros (list of commands), and for undo and redo management.

Once retrieved, commands related to each user component are put in menus or toolbars by the User interface component.

Figure CAPRA.3 shows an example of how the communication between the user interface and the IGES reading component works. IGES (Initial Graphic Exchange Specification) is a standard format for exporting mathematical description of parts from CAD applications. A CAM software needs to deal with files of this and of other standard formats in order to produce commands for the machine that will manufacture the part. Two example operations are checking compliance of a file with IGES and reading the file: they are represented via the `CheckIGESCommand` and `ReadIGESCommand`.

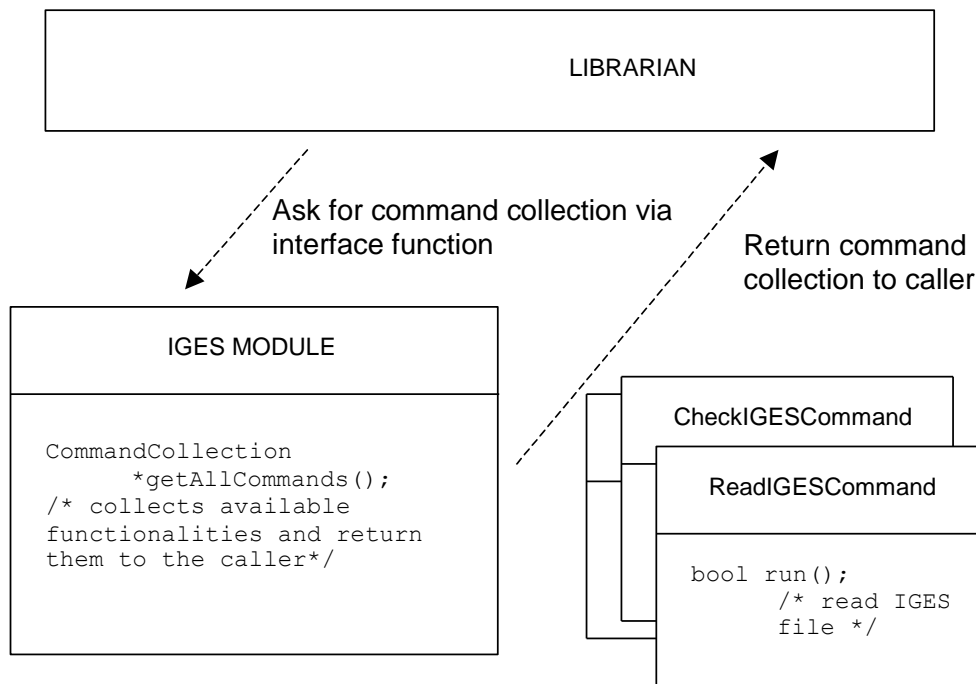


Figure CAPRA.3. Communication example.

The IGES component returns a collection of commands to the librarian component. As the user clicks the menu item corresponding to the reading of an IGES file, the methods for asking parameters and execution of the corresponding command are called (`run` in Figure CAPRA.3). In the example a dialog box asking the user to choose the name of the file is displayed (ask the user) and then the mathematical description

contained in the IGES file is converted in the internal representation (execution).

Base components that do not export functionalities directly to the user return a void collection.

Communication component - component

Components will often have to cooperate to perform a task, so there is the need to have them communicating with each other: they are requested by the librarian to register their services as CORBA compliant objects. Each service is derived from a specific CORBA IDL, defining its interface. The librarian component fills a map with the name registered by the server and the server object, and acts as a connector: a component requires a service via its registered name, and gets back the object exploiting this functionality.

An example of registration for curves offset computation is represented in Figure CAPRA.4.

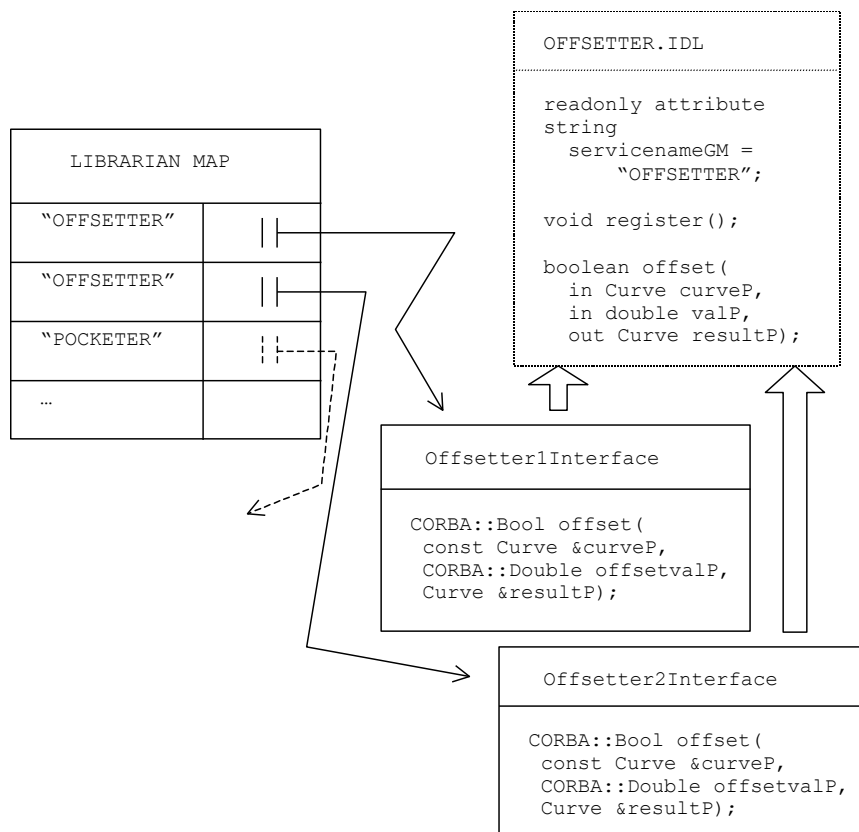


Figure CAPRA.4. Inter-component communication.

Figure CAPRA.4 shows an example of librarian map where 2 servers compliant to the OFFSETTER.IDL are registered. The specification defined by the IDL, and registered to the librarian as "OFFSETTER", is the computation of the offset curve of a given bidimensional curve.

A smart mechanism of retry can be implemented if more than one server is registered

with the same name: if the first offsetter fails in offset computation, the client component may ask the librarian if another “OFFSETTER” exists, and try to use this second object for offset computation.

In the example, librarian map contains another registered server (“POCKETER”), not represented in the figure.

After the map initialisation phase, during modules loading, librarian uses it as a connector between the component requiring an “OFFSETTER” and the object that has registered this interface, in this case the object computing the offset.

Component repository

We use CVS as component repository. In CVS the lower level configuration item is a file. Directories can be used to structure files. A starting directory containing the structured source files for a module (in no-component programming it is often the whole application), is imported into the repository as a CVS module (the higher level configuration item).

In our case a module is implemented by several source code files. A module is sold to a customer as a full fledged product, so we need to keep track of its evolution over time, with several different versions. For corrective maintenance we need to be able to rebuild a product starting from any version of any module.

We map a module to a directory in CVS. This allows to track different versions of a module, independently from other components.

Further, we use the CVS scripting language to define retrieval and assembly of configuration items, thus eliminating chances of errors.

5. Conclusion

We have presented the rationale and the main design choices for an infrastructure that supports development, maintenance and distribution of components in the context of a CAM product. We use CORBA as support for components, and CVS as tool for the repository of components.

Currently, around 80% of the HI-MILL and ISOGRAPH products have been redesigned with the component approach, using the infrastructure presented in the paper. The main lessons learnt are:

- The design of the product, using the component based approach, is more complex. Especially the use of the Command pattern is not easy to every software engineer: the problem is that object oriented philosophy applied to framework is very different from component based object orientation, addressing sometimes opposite issues. As a result developers are having difficulties in this part. We believe this a learning problem that will eventually be solved.
- The new design of the product is quite stable from the beginning. In other words, it expanded but not changed too much from the initial sketch. This was quite surprising, given the complexity. We think the reason lies in the availability of proven templates (like patterns published in literature) that can be applied quite

flexibly to different domains, like CAM software.

We are also monitoring with care the following issues:

- Performance. The communication between components requires an overhead, that may not be acceptable from a performance (execution speed) point of view. The link with the Numerical Control component, subject to real time constraints, is particularly sensible.
- Evolution of components and compatibility. The evolution of components over time may break the requirements on compatibility of interfaces.

We are going to experiment the new approach with a few customers in the next phase of the project. Here we will monitor closely technical difficulties, and the acceptance of the new approach from the clients.

Acknowledgements

This work was partially supported by the European Commission, under contract IST Reindeer n. IST-1999-20288.

References

- [1] O.P. Brereton et al., The Future of Software: Defining the Research Agenda, *Comm. ACM*, Dec. 1999, pp. 78-84.
- [2] O.P. Brereton and D. Budgen, Component-Based Systems, a Classification of Issues, *IEEE Computer*, Nov. 2000, pp. 54-62.
- [3] A.W. Brown and K.C. Wallnau, The Current State of CBSE, *IEEE Software*, Sept./Oct. 1998, pp. 37-46.
- [4] CORBA, www.corba.org
- [5] CVS, www.cvshome.org
- [6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns*, Addison-Wesley 1999. Author Biography
- [7] C. Szyperski, *Component Software—Beyond Object-Oriented Programming*, Addison Wesley Longman, Reading, Mass., 1998

Session 12 - SPI and European Results and Benchmarks

**Session Chair:
Risto Nevalainen, STTF, Finland**

SPIRAL Network development Results	12-2
PIPSI – The Results of a Two Year SPI Experiment	12-19
Web based SPI – Benchmarking of Software – Organisations	12-33

SPIRAL Network Development Results

Béatrix BARAFORT

*Joint contribution of the SPIRAL*NET¹ team*

Centre de Recherche Public Henri Tudor, L-1359 Luxembourg

Introduction

SPIRAL*NET was an ESSI ESBNET project[1] (European Systems and Software Initiative of the European Commission supporting Software Best Practice Networks) aiming at **optimising and generalising best practices in the Customer/Supplier processes quality management** (CSPQM). The targeted activities were Customer/Supplier processes, support processes such as quality assurance, joint reviews, configuration management, documentation management, project management.

The SPIRAL*NET project allowed the development of a network (nominated SPIRAL) in a French speaking area composed of the Grand-Duchy of Luxembourg, Wallonie (the French speaking part of Belgium) and the French Lorraine. The objectives of the project were :

- to make the market aware of best practices on Customer/Supplier processes and associated support processes,
- to improve the visibility and the access to structured information related to the regional software market,
- to provide and standardise the market with CSPQM tools selected from best practices and adapted to regional practices,
- to support CSPQM implementation with the shared tools,
- to provide the market with service offers on CSPQM (services supported by the project partners and a qualification programme proposed by the project).

The partners of the project were the Centre de Recherche Public Henri Tudor in Luxembourg as co-ordinator, the Centre de Transfert de Technologie de Charleroi (innovation Centre created from the University of Namur – Facultés Uuniversitaires Notre-Dame de la Paix) in Belgium and the “Unité de Formation Recherche – Mathématiques et Informatique” of the University of Nancy 2 in France. They developed an implementation strategy throughout 5 layers in order to meet the objectives :

- to heighten the market awareness,
- to set up a common electronic platform,
- to select and to share support tools,
- to support the implementation of CSPQM and the use of common tools,

¹ SPIRAL*NET is the ESSI ESBNET project 27884

- to provide the market with qualification and certification.

Implementing several activities by following this 5-layer underlying strategy was the mean to progressively fulfil these objectives. This paper describes how the objectives of the project have been reached[2][3] up to the end of the project, and how the SPIRAL network was developed throughout these activities.

Market awareness

One of the project's objectives was to make the market aware of best practices on Customer/Supplier processes and associated support ones. IT professionals, specially small software development teams and SMEs/SMIs have to be made sensitive to the benefits of a formalised approach and of Customer/Supplier quality processes through standardised tools and dedicated support services. In order to do so and to promote the SPIRAL network development, several actions had been organised like awareness events and workshops, participation in local trade fairs and exhibitions, and prospective visits.

Kick off meetings, awareness events and fairs

Awareness meetings have been organised in October 98, in both sites of Luxembourg (L) and Namur (B). In Luxembourg, the Kick off event occurred during the SPIRAL'98 annual conferences. These meetings were the opportunity to officially launch the SPIRAL network and to announce next proposed activities for 1999. Each of the above mentioned awareness actions was the opportunity to distribute SPIRAL documentation. There were also complemented by articles published in local and regional newspapers and SPIRAL stands were exhibited in local trade fairs.

In March 99 , in June 1999 and in March 2000, respectively in Luxembourg, Charleroi (B) and Metz (F), a specific event has been organised in order to launch working groups on topics such as Project Management, Customer/Supplier relationships, Software Engineering. All events were the opportunity to promote and remind the attendees of the SPIRAL activities.

Prospective visits

Prospective visits aimed to aware companies on CSPQM, to attract them in the network, to take part in activities and more particularly to include them in the software regional directory. A co-operation agreement has been developed and companies were asked to sign it in order to register and formalise their membership in the network.

Considering all contacts established through the awareness actions, prospective visits were systematically planned in order to meet IT managers in banks, insurance companies, software houses, institutions and administrations.

The SPIRAL network activities were promoted and the following actions and/or activities were proposed :

- to register and formalise the membership in the network,
- to appoint a quality interlocutor (from the SPIRAL*NET project team) for the company,
- to register and provide descriptive information about the company's IT activities and competencies in order for the company to be listed in the regional software directory,

- to make the company know the training courses available in the SPIRAL training catalogue,
- to participate in working groups,
- to be assisted and advised in Software Process Improvement (SPI) (SPI services were proposed such as micro-assessments, SPICE assessments, improvement plan determination, SPI actions implementation, customer/supplier relationships assistance),
- to participate in a specialised training cycle in SPI.

All the project long, 120 companies were met individually.

For each one, a meeting has been planned and a personalised presentation prepared. A presentation kit had been built and was used during the meetings.

80% of the companies met were SSII, 15 % were IT departments and 5 % Administrations or Public Institutions.

Common electronic platform

In order to attract as many IT actors as possible in the SPIRAL network and to provide on-line facilities, a common electronic platform has been prepared at the very beginning of the project, with basic facilities. Then, an electronic frame has been set up in order to support all activities related to the SPIRAL network.

A dedicated Extranet

The SPIRAL electronic platform (SPIRAL web site: www.spiral.lu) proposes the deployment of several Telematics services dedicated to IT professionals. The main on-line services are the SPIRAL training course offer (on-line catalogue), the regional software market directory and a work placement market. Associated to these services, the platform proposes a co-operative space enabling the sharing of information and on-line discussion in the form of forums and working groups.



Fig. BB.1 : The SPIRAL Web platform

A login is provided for SPIRAL members (with authentication mechanisms) in order to offer them all SPIRAL services and "added value" information, whether public visitors have restricted access within the SPIRAL web site. A "*Qui sommes-nous*" link presents the SPIRAL partners (with the SPIRAL*NET project summary).

Several entry points are provided through the left frame. Firstly the ones about the SPIRAL Network :

- **Mission**
Promoting quality and innovation related to information systems management, strategic and engineering practices.
- **Charter**
Quality and innovation commitment. Document signed by all partners
- **Membership conditions**
To join the network, to meet the Charter requirements and to pay an annual fee
- **Membership**
To fill on-line membership forms.

Secondly, the entry points about the SPIRAL Services :

- **Training Tele-market**
To access to the SPIRAL training catalogue, to provide facilities for the members for capturing information about their training offer and to publish it on the same way as the SPIRAL catalogue.
- **Conferences**
To announce local , regional and international conferences.
- **Thematic meetings**
To announce thematic meetings gathering IT professionals wishing to discover /discuss about technical and methodological topics related to best practices in the domain.

- **Working groups**
To present actual and future working groups on topics such as Project Management, IT Security (with the CLUSSIL CLub Utilisateurs pour la Sécurité des Systèmes d'Information Luxembourg) and Luxembourg Java User Group (LUXJUG).
- **Regional software directory (Competencies market)**
Cf. paragraph 3.2.
- **Support to innovation**
To provide services such as micro-assessments, SPICE assessments, coaching and advising for the re-engineering of the information system., coaching and advising for supplier selection.
- **Work placement market**
Cf. paragraph 3.2.
- **Networks of partners**
To explain and provide links to other networks such as the CLUSSIL, the LUXJUG,...
- **Request for proposal market**
To provide facilities for posting requests for proposal.
- **Documentation Centre**
To provide facilities for consulting and downloading SPIRAL*NET technical deliverables such as methodological guides, document templates, case studies,...
A search engine related to the whole web site is provided. A newsletter and a contact with the SPIRAL team via an e-mail link are also provided.

SPIRAL Extranet Architecture Remark

The Telematics services have been designed in order to be generic and reusable. The will of using innovative technologies in the Telematics field led the software developments to domains such as object-oriented databases, automatic publication of data, XML. Telematics services are available with authentication mechanisms for SPIRAL members, and particular services.

The directory of the regional software market (competencies market)

The main component of the electronic platform is the directory of the regional software market. This directory is aiming at providing structured information on IT suppliers and customers. So that, for instance, buyers are able to electronically consult on their needs. The directory shows IT services and competencies of the Lorraine, Walloon area, and Grand Duchy of Luxembourg.

The displayed professionals are IT departments, software houses, IT independent workers, associations and individual members. The directory is aimed at people seeking representative and precise information on one or more actors of the software market and competencies or services meeting their needs. And this is happening in very various contexts such as a subcontractor or partner selection in the scope of a request for proposal, or the seek for specific competencies or information for a platform or environment use in a given framework.

After a first development phase, a prototype (with an Access database and Microsoft Active Server Pages technique) has been put online at the beginning of January 1999. It was accessible to general public through the SPIRAL Web site. In its first version, it was

aiming at gathering information (descriptive information, activities and references) about companies of the SPIRAL network targeted area. This service allowed the search for companies via a multiple criteria search engine. Then it enabled the information retrieval and consultation concerning the result of this research.

The prototype reached its limits in terms of performance, data storage and lack of IT competencies function. The integration of the companies' competencies was the next goal to reach in the project in order to improve the visibility and the access to structured information related to the regional software market. So, a second phase of the directory development was devoted to the analysis of the context (definition of the IT competencies structure, needs analysis, alternative solutions study, final solution choice), the implementation (design, tests of the tools, technical implementation), and extended specification has been written in order to develop and offer on-line forms to the companies, directly from the directory web site. These forms allow to collect all type of information related to a company : descriptive, activity and IT competencies.

Since the end of the project, the 3.0 version of the directory is available from the SPIRAL web site (with 46 registered companies and 355 registrations from a public file).

The work placement market service

The market of the work placements is another service on the SPIRAL Web site that can offer three function levels:

- a **companies** function, thanks to which companies can publish work placement offers,
- a **higher education establishments** function, thanks to which a profile for work placements can be presented,
- a **students** function, thanks to which students can post their candidatures.

For each function, it is possible to consult work placement's offers and demands.

This service is in relation with the directory of the regional software market previously described. As for the directory, the work placements market service covers the SPIRAL geographical area, but it can be extended to other regions and countries. Actually, this Telematics service only covers the IT field. In the future, this service could be spread out to other fields than the IT one.

The first functions were developed with the Java platform, Internet technology and an Object-Oriented database. The last function used this environment, plus XML technology.

The Telematics forum

A tool named AltaVista Forum 98, accessible by a Web browser complemented this electronic platform. The AltaVista Forum is an application that provides an easy way to communicate and to share resources with different groups of people.

Selection and sharing of support tools

Main topics structuring the SPIRAL*NET activities development

In order to provide and standardise the market with CSPQM tools, the SPIRAL*NET project partners identified main topics throughout the SPIRAL network. These are the following :

- Project Management,
- Customer/Supplier relationships,
- Software Engineering.

*A technology watch was made on these topics. All elements related to them were considered in order to contribute to the preparation of SPIRAL activities and to provide the electronic platform with quality tools, models and guidelines such as case studies, framework models, questionnaires, reference sets, and recommendations. A Project Management tool has been particularly studied for this period and was experimented in the SPIRAL*NET project itself. The three above-mentioned topics also kept the thoughts going with requirements for working groups.*

All the approach was supported by a methodological set stemmed from the "Process Professional Portfolio" (PPP)[4] licensed by Compita Ltd (UK) and by ISO/IEC 15504 standard[5][6].

Working groups

The working groups were a meeting point and an experiment exchange space between IT actors willing to improve their professional practices and to contribute to the IT profession enhancement. These meetings gathering peers focused on the three targeted quoted topics. Participants exchanged viewpoints, shared their analysis and experiences, invited experts and thus participated in the development and the animation of the SPIRAL network. Each working group set its goals and expected deliverables. It was through these working groups that recommendations and harmonisation means for best practices started to be established

There were three organised working groups in Luxembourg, focusing on the improvement of software engineering practices, project management practices and customer/supplier/ relationships. A fourth working group was organised in Namur (Belgium) and covered the 3 previously mentioned topics. Unfortunately, there were not enough participants for organising a working group in the French Lorraine. Despite numerous awareness actions, this part of the SPIRAL region did not reach the expectations and the SPIRAL*NET project goals for participating in such activities.

The first half of the SPIRAL*NET project was devoted to define, prepare, promote and launch working groups, the second one to run them. Until the end of the project, 25 sessions among all working groups occurred. Up to the end of the project, 63 people were registered for the 4 working groups (participants, speakers and SPIRAL*NET team member included), that is to say 42 companies; 63 registered members from regional companies actually participated in one session or another.

Most of the time meetings were organised with the following outline : theoretical presentations, case studies presentation, debate / discussion, thoughts about the deliverable to produce on the studied topic. Theoretical and case studies presentations transferred information to participants who directly benefited from them. They triggered discussions about current practices within the participants companies. These discussions

were rather rich even though not structured. And because of lack of time, deliverables thoughts rarely succeeded.

At the end of the project, the statement was that the meeting agendas were often too ambitious. So the deliverable thoughts that were planned could not be performed properly.

The working groups success was measured through the participants' assiduity and the produced deliverables. For the first sessions, the objective was to mainly win the "loyalty" of the registered participants. Then the sessions were more focused on theoretical presentations than on deliverables production. Even if there were no proper deliverables produced, it was a very rich, rather rough set of material.

Because of several changes in the SPIRAL*NET project team, it was rather difficult to maintain the same frequency of working groups sessions during the year 2000, and the number of participants also lowered. In order to properly formalise the project deliverables, there were no working group sessions organised at the end of the year 2000 and at the beginning of 2001, but with an end-of-project-workshop organised in April (in order to present SPIRAL*NET project results and SPIRAL network activities), there was the opportunity to launch new series of meetings, based upon SPIRAL*NET working group experiences. Two types of actions are considered :

- Working groups with real involvement of registered attendees and the clear objectives to produce deliverables, on targeted topics,
- Thematic meetings will be the opportunity to share information throughout experts presentations, with objectives like awareness, dissemination and demonstration of IT competencies.

SPIRAL*NET methodological toolbox

*In order to provide the SPIRAL network with a toolbox composed of best practice materials, methodological guides and case studies, the expected deliverables have been produced by the SPIRAL*NET team members, on the basis of all gathered material such as the working groups one, but also with the training course handouts, the mentoring and support services experiences with all associated reports (case studies). Considering the 3 previously mentioned main topics, the methodological toolbox was built according to a process approach. A set of processes were particularly targeted :*

- Acquisition
- Requirements elicitation
- Project Management
- Software Process Assessment and Improvement, with 3 nested assessment sub-processes :
 - Micro-assessment
 - OWPL assessment
 - SPICE assessment

Software engineering processes were not particularly described throughout a specific methodological guide, but the Software Process Assessment and Improvement guide is the structured approach to use for improving such processes. And working groups results related to Software Engineering were used for writing this guide.

The framework that combines all these deliverables is the following :

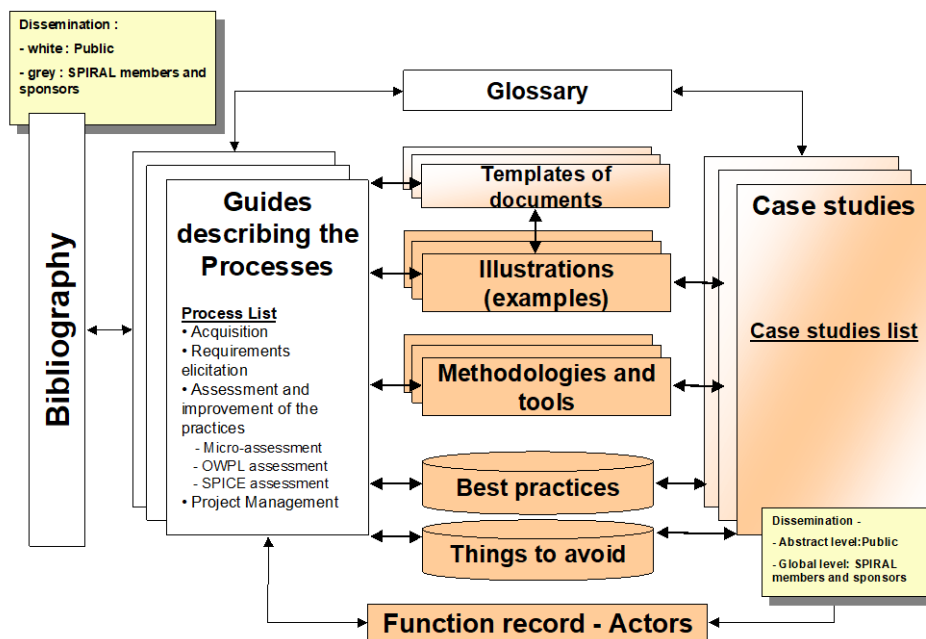


Fig. BB.2 : The SPIRAL*NET methodological toolbox

These deliverables constitute the main results of the SPIRAL*NET project.

All the guides describing the processes follow the same structure :

- introduction
- context and process approach
- for each process activity : description with goals, inputs, outputs, actors, activity progress, used tools and methods, examples.

Each guide has links towards associated best practices and things to avoid, document templates, examples, and sometimes a short description of methodologies and tools. Among case studies, there are experimentation of processes described within guides, and/or only some activities of these processes. The examples can be check-lists, descriptions of activities of a process in a particular context, prepared questionnaires for assessments,... Document templates can be quality plans, request for proposal, assessment report,...

There are general documents related to all deliverables : glossary, bibliography and function record for actors.

For consulting these deliverables, cf. the SPIRAL web site : Documentation Centre. The documents in “white” are for public access. The other ones are only accessible for SPIRAL network members.

Implementation of CSPQM processes and use of common tools support

The implementation consisted in providing the network with a direct support in launching CSQPM and their associated tools in business relationships. Training courses were organised, support services were provided to companies throughout mentoring projects such as :

- assistance in defining an IT strategy,
- assistance in the IT architecture design based on new technologies,
- coaching and advising for the re-engineering of the information system,
- coaching and advising for supplier selection,
- coaching and advising for supplier follow up during the solution implementation,
- software quality standards awareness meetings,
- improvement quality project definition in an ISO 9000 certification preparation context,
- assistance in project Management and quality assurance for project activities.

All mentoring actions and support services were SPI experiences and CSPQM implementations in the SPIRAL network. They were progressively formalised in order to contribute to a case studies corpus. Each case study adopts the same structure: summary, context of the firm, of the actors and project description, project running (description of project stages), process points of view, methods and tools points of view, project results : results compared to the initial objectives, identified strengths and weaknesses, lessons learned.

Micro-assessment

A micro-assessment service has been developed and performed within several companies. A company contacted within the SPIRAL network can be put through the questionnaire by telephone. It addresses the vision of the company and the CSPQM practices. Results of the questionnaire point out the planned improvement actions and their effects in the company.

In the assessment context, regular contacts and sharing of experiences have been established with a project team working for the Walloon region on the building of a SME dedicated framework for software assessment (via a project named OWPL for "Observatoire Wallon des Pratiques Logicielles"). Common work has been accomplished for the micro-assessment development.

The aim of the micro-assessment was to give a first outlook of the software practices in an organisation, to make a diagnosis and guide the next steps of software process improvement. The main requirement that drove the design of this model was to be as less costly as possible, in time and money.

So, the designed model corresponds to an half an hour interview based on a well-prepared questionnaire. The questionnaire covers six key lines selected as the most pertinent and the most prior to target organisations. The questionnaire has been based on SPICE and CMM concepts[7]. All adaptations were made by considering that the micro-assessment had to be short in time and money, and had to particularly suit SSDs and SMEs. So the key lines are the following:

- quality assurance,
- customers management,
- subcontractors management,
- project management,
- product management,
- training and human resources management.

31 micro-assessments were performed during the project running (IT small companies, IT services in other businesses, public administrations using IT). The experience showed that the micro-assessment model was very attractive for SSDs and

SMES as a tool to start with SPI, mainly because of its extreme simplicity. As the assessed units were not totally representative of the SPIRAL area IT market, no general conclusion about this study could be made.

The micro-assessments that were carried out have shown that there was not one common weak point in all the assessed units. Moreover, each of these units had its own expectations towards this SPIRAL*NET action and the long-term results of this assessment. Their answer to the SPIRAL invitations, their participation to awareness events, and their agreement to take part to an assessment of their software practices were some good revealing elements. The study especially reinforced, through the wishes which have been expressed by the units, the convictions as for the utility of the actions that have been carried out during the project and their relevance with respect to expectations of the actors of the IT market.

According to that element, assessments of this type will be integrated in the SPIRAL membership model and provided within member units and, in parallel, the follow-up of the first assessments will consist in the repetition of annual evaluations, allowing to note the evolutions and improvements already brought

CSPQM qualification and certification

In order to ensure continuity of the SPIRAL proposed services, the network developed a qualification and certification process. The global guidelines of the process are provided within the "SPIRAL Charter".

In order to contribute to this process, a training cycle dedicated to SPI had been defined. It was named "Amélioration des Pratiques Logicielles" (APL) for improving software practices.

This training cycle aims at producing quality engineers in the Information System (IS) field. At the end of the cycle, attendees are able to implement a software quality process in their company after :

- initiating a software practice improvement programme,
- deploying and implementing quality assurance activities in Information System projects,
- mastering a methodological baseline and competencies in order to combine business, organisation and the company's strategy with its improvement goals.

The training cycle addresses anyone involved in a quality process in an IT department or a software house.

*The cycle alternates theoretical sessions, case studies and experiments analysis with practical actions within each attendee's respective company. An on-site monitoring is proposed. The unifying thread of the cycle is the software practice improvement project adapted to the company's goals and specificity. The dynamics induced by the control of actual quality actions enables the training cycle participants to gain concrete experiences. The cycle progress was organised with 10 monthly sessions interspersed with on-site assistance dispensed by a SPIRAL*NET team member.*

Training cycle structure

The SPIRAL*NET SPI training cycle had been build according to a progressive

approach. Guidelines of the ISO/IEC 15504 standard recommend 8 steps in the "Guide for use in process improvement" (Part 7). The adopted approach for the APL training cycle was based on these SPI steps, gathered in 4 main stages. For each of these stages, one or several training sessions were the opportunity to tackle associated concepts, and to explain appropriated tools and techniques. This was aiming at progressively building the methodological framework that was the main outcome of the cycle and to give the participants all necessary components in order to implement improvement actions within their company, all along the training sessions.

APL training cycle results evaluation

The training sessions occurred from July 99 to November 2000, with a final exam in December. A professional certificate was delivered to the attendees in April 2001, during the end-of-project workshop.

All participants were given satisfaction with the training cycle running and contents, despite the fact that it lasted longer than previously planned because of availability problems of the trainers. It was also difficult for the participants to run improvement actions within their companies for different reasons such as :

- lack of support from their sponsor (manager),
- some of them changed from a company to another, or from a department to another within their company,
- the selected process(es) improvements were sometimes stopped, due to critical events in their company, with the main known and common one : the Y2K passage.

This training cycle was proposed in the SPIRAL training catalogue 2001, with a few adaptations :

- the training cycle duration was limited to 6 months and 6 sessions (it is particularly important because of the very high staff turn-over in most IT companies at the moment)
- the training programme was reduced from 10 sessions to 6 with focus on main relevant topics quoted by the participants of the first version of this training cycle
- optional assistance days for participants were proposed, instead of systematically providing them in the training package, with a request for formal support by a sponsor.

Conclusion

For the first half of the project, a great interest has been recorded for the SPIRAL network in connection with the harmonisation of the market software practices, with the acknowledgement of SPI initiatives, approaches, and competencies in the regional companies aiming at improving their practices. Moreover, this interest has been shown despite a strong mobilisation of human resources in the companies for Euro and Y2K projects, and a high staff turnover in most of the contacted companies since the beginning of the project.

The second half of the project consisted in implementing all prepared activities and in gathering as many actors as possible within the network. Acknowledgement mechanisms by the IT professionals themselves, and the maintenance and means to perpetuate SPIRAL*NET activities after the end of the project were also studied.

Dissemination is dealing with the promotion of the network and wide dissemination of the SPIRAL*NET results on a national and regional scale, and throughout all regions of Europe. To do so, several dissemination actions occurred all along the project :

- SPIRAL*NET team presentations in SPIRAL annual conferences, in “La journée luxembourgeoise de la Qualité – 2000” conference in Luxembourg, in a conference named “Vers une maîtrise de la qualité logicielle” in Charleroi (B) in June 2000,
- Meetings organised with the FESPINODE project (French ESPINODE) in the three sites of the SPIRAL area,
- dissemination materials provided by European projects : SPIRE[8] (ESSI Project 23873), SMILE[9] (ESSI Project 23973) and distributed all along SPIRAL*NET activities,
- papers presented in European conferences in Software Process Improvement (EuroSPI'98[10], SPI'98[11], EuroSPI'99[13], SPI'99[14][12]),
- an end-of-project workshop named "Vers l'Excellence des pratiques d'Ingénierie des Systèmes d'Information" was organised on April 24th in order to present all SPIRAL*NET project deliverables (the methodological toolbox in particular) and the way the SPIRAL network activities are going to carry on after the official ending of the project (a new project is being defined in CRP Henri Tudor with the collaboration of the FUNDP and the Infopole in Namur (B), in order to perpetuate current activities and to develop new ones)

After the end of the project, all the opportunities to promote and disseminate SPIRAL*NET project results will still be grasped on a national, regional and international scale.

The elaboration of a membership strategy began in January 2000 with a principal purpose : to create an open model sustaining the collaboration between the “SPIRAL” network actors. The development of this model also allowed a reflection and a focus on the essential mission of the spiral network : the promotion of Quality and Innovation best practices in software engineering. The principal adjustments were related to the scope of the offered package of services and the associated cost. Indeed, the membership model was to be accessible for all kinds of members, not only the most mature (in process improvement) or the biggest or most profitable.

The result membership strategy defined and validated by the SPIRAL*NET project partners considered two types of actors in the network : **members** (adhere and respect the criteria of the SPIRAL CHARTER OF QUALITY, pay an annual fee) and **sponsors**.

In addition to the SPIRAL products and services, a Member or Sponsor, associate its image with that of a network of Excellence, neutral and independent and take profit of all the repercussion of activity of the Network. All the new web site (Cf. chapter 3) was built around this model and the associated products and services.

All opportunities for developing SPIRAL*NET project activities have been studied in order to limit or if possible to cancel risks about the impact of the project which was different in the Walloon area, and more particularly in the French Lorraine, compared to Luxembourg. The project partners there (particularly in France) have university missions which prevent them from participating as actively as the others in the SPIRAL*NET activities. Furthermore, the software regional context in French Lorraine is much less dynamic than in the other SPIRAL areas. Despite all performed effort, at the end of the project, its impact remains different in each part of the SPIRAL area because of the above mentioned reasons. This also reflects the IT maturity level in these areas. Awareness actions still have to be organised in order to equalise, and above all to raise

this maturity level, and to lead all IT professionals to join the SPIRAL network.

Finally, the perpetuation of the SPIRAL network activities will start throughout the implementation of the previously presented network model which can attract and include all kinds of IT professionals. It will insure the network animation actions in all the SPIRAL area after the ending of the SPIRAL*NET project.

References

- [1] SPIRAL*NET Project Programme – ESSI Project 27884 – V 2.0, May 1998
- [2] SPIRAL*NET Mid Term Report – ESSI Project 27884 – V 1.0, June 1999
- [3] SPIRAL*NET Final Report – ESSI Project 27884 – V 1.0, April 2001
- [4] Compita Ltd, Process Professional Portfolio, Process Professional Library Services, 1996
- [5] XP ISO/IEC 15504, Parties 1, 2, 3, 4, 5, 6, 7, 8, et 9 : Septembre 1998 et Partie 5 : Décembre 1998, Technologies de l'information – Evaluation de processus de logiciel, Edition AFNOR en français du référentiel ISO/SPICE
- [6] EN ISO 9001, Systèmes qualité – Modèle pour l'assurance de la qualité en conception, développement, production, installation et prestations associées, CEN, Bruxelles, B, Juillet 1994
- [7] Dymond K.M., Le guide du CMM (SM) – Introduction au modèle de maturité CMM, Traduction : A.Combelles et al. – Objectif Technologie, Cépaduès Editions, Toulouse, F, 1997
- [8] The SPIRE Handbook : Better, Faster, Cheaper Software Development in Small Organisations – ESSI Project 23873, 1998
- [9] SMILE (Spreading Multimedia Information for Learning and Enlightenment about Software Process Improvement) CD-Rom, Multimedia information system and CD-Rom funded by the ESSI Project 23973, 1998
- [10] Barafort B., Hendrick A., A Small Software Developers' framework for evaluating the internal usage of IT, in: *Proceedings of the International Conference EuroSPI'98*, Göteborg, Sweden, 1998
- [11] Barafort B., Hendrick A., A SME dedicated framework to evaluate the internal usage of IT, in: *Proceedings of the International Conference Software Process Improvement*, SPI'98, Monte-Carlo, Principauté de Monaco, 1998
- [12] Habra N., Niyitubabira E., Lamblin A-C, Renault A., Software Process Improvement in Small Organisations Using Gradual Evaluation Schema, in: *Proceedings of the International Conference Product Focused Software Process Improvement*, PROFES'99, Oulu, Finland, 1999

- [13] Barafort B., Hendrick A., Mid-term results of the SPIRAL Network Development, in: *Proceedings of the International Conference EuroSPI'99*, Pori, Finland, 1999
- [14] Habra N., Niyitubabira E., Lamblin A-C, Renault A, Software Process Improvement for Small Structures : First Results of a Micro-Assessment framework, in: *Proceedings of the International Conference Software Process Improvement, SPI'99*, Barcelona, Spain, 1999

We would like to thank all SPIRAL team members who contributed to this paper.

CV of the author

Béatrix BARAFORT

Has worked in a software house in Lyon (France) for 8 years :

- Programmer, software analyst, and then project leader for development projects in banks and insurance companies.

Joined the Centre de Recherche Public Henri Tudor in Luxembourg at the end of the year 1996 for working in software process improvement projects :

- Co-ordinator of several projects of process assessment (SPICE) and improvement programs,
- Project leader of the SPIRAL*NET ESSI ESBNET Project 27884,
- SPICE Qualified assessor (certificate of achievement of “Process Professional Assessment”).

Centre de Recherche Public Henri Tudor

The Centre de Recherche Public Henri Tudor, founded in 1987 as a public research centre, was created to promote innovation and technological development in Luxembourg. The Centre's goal is to improve the innovation capabilities of the private and public sectors by providing support services across the main technology-critical areas : information and communication technologies, industrial and environmental technologies. It is assisted in its mission by a diversified network of industrial and institutional partners.

The Centre de Recherche Public Henri Tudor participates in European Union programmes including ESPRIT, Craft, Info 2000, LIFE and Telematics Applications Programme. As a result, Luxembourg businesses are able to draw on the knowledge and expertise of Europe's greatest research centres.

The Centre is also actively engaged in inter-regional co-operations within the "Grande Région" (Saarland and Rheinland-Pfalz in Germany, Lorraine in France and the province of Luxembourg in Belgium). It is a co-founder of the European College of Technology, a tri-state initiative based in the European Development Pole at the Athus-Longwy-Rodange intersection, and contributes to the innovation programmes of the EU Structural Funds.

Main figures

- a full-time staff of 160
- 5 research laboratories
- 4 innovation support services
- 6 technology resource centres
- annual turnover of more than ECU 6 millions
- 60 % self-funding

PIPSI – The Results of a Two Year SPI Experiment

Rory O'Connor
Dublin City University

Gerry Coleman
Dundalk Institute of Technology

Introduction

Software engineering is a discipline whose aim is the production of fault-free software that satisfies the users needs and that is delivered on time and within budget. Sounds nice, but in order to achieve these goals, appropriate software engineering techniques have to be used in all phases of software production, and maintenance. In order for these techniques to be employed, they have first to be taught. Once in use, they may then be integrated as part of the corporate culture and improved upon.

The training provided should enable the definition of a software process that supports sound software engineering principles. Individuals should be trained in the necessary techniques, which facilitate process maturity. The training should be geared toward the perspective of the individuals being trained for it to have the most effect. For example, it is not necessary to train an entire set of developers in the nuances of Software Quality Assurance (SQA), it is enough that they understand the role of SQA.

All members of the team should be provided training in the applicable life cycle model. This should include the various phases of the software process, from requirements to retirement, paying special attention to problems associated with each phase. This type of training provides all members of the team with a common understanding and clear indication of the road to completion. These individuals should also be provided training in any applicable new technology areas. For example, training in object-oriented design and/or software reuse.

Technology can help to tackle many of the problems associated with software development, but ultimately people and organisations produce software. This straightforward observation has led to focus improvement on the software process. From this point of view, we can recognise two intertwined trends: Firstly, software process improvement at the organisation level, such as CMM, the ISO 9000 series and SPICE (now ISO 15504). And secondly, software process improvement at the level of the individual or small teams, such as PSP, PIPSI and TSP [1].

Process improvement at the organisational level suffers from some drawbacks. It is usually a very top-down approach, that gives little support to individuals and risks not addressing their day to day problems; and it is cumbersome and especially ill-adapted to SMEs. The bottom line is that people develop software. Establishing, a software engineering process, and improvement upon that process, depends on the individuals working in an organisation. If the individuals are disciplined, and adequately trained in the principles of software engineering, the chances are good that the software process will be successfully defined, implemented, and improved. This, in-turn, increases the probability of achieving the aim of an on time, within budget, error-free software product.

In the following two sections we introduce two SPI methodologies aimed at the level of the individual software engineer, the PSP and PIPSI, in order to get a better perspective of individual software engineer's SPI needs. We will then compare the results of these two methodologies and assess both their usefulness and relevance in the context of the European software engineering business.

PSP

The PSP (Personal Software Process) was developed at the Software Engineering Institute (SEI) by Watts Humphrey. It is designed to bring discipline to the practices of individual software engineers. The PSP provides students and practising software engineers with a framework for measuring and analysing their development work so that they produce programs of higher quality and in a more predictable manner. More specifically the objectives of the PSP are as follows:

- To introduce students and engineers to a process-based approach to developing software
- To show students and engineers how to measure, estimate, schedule, and track their work
- To show students and engineers how to improve the quality of their programs

The concepts, structure, and activities of the PSP are described in detail in the textbook “A Discipline for Software Engineering” [2] in which Humphrey characterises the purpose and scope of the PSP: “The PSP’s sole purpose is to help you be a better engineer...It can help you plan, better track your performance precisely, and measure the quality of your products. Whether you design programs, develop requirements, write

documentation, or maintain existing software, the PSP can help you do better work...The PSP is not a magic answer to all your software problems. Although it can suggest where and how you can improve, you must make the improvements yourself” [2].

Components of the PSP are not complicated and they are based on sound engineering principles, both teachers, students, and engineers have found that learning PSP is a demanding and challenging activity. PSP training is a significant investment (about 150 hours per engineer to complete the course). This is because the course is more than just a class; it is a boot camp. It goes beyond telling the engineers what to do, by having them use the principles while writing ten programs and collecting and analysing data about their performance. In the PSP training, the engineers convince themselves of what works for them and what doesn't so that they can take control of their personal software process.

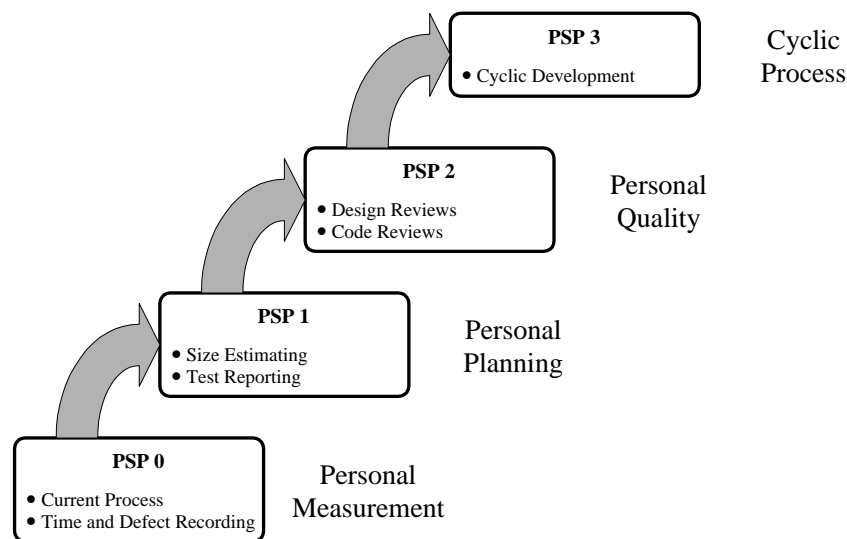


Figure 1 - The PSP Structure

The PSP provides an incremental approach. It includes seven PSP processes, grouped into four process levels (figure 1), that have following focus:

- PSP0 - establish a measured performance baseline
- PSP1 - make size, resource, and schedule plans
- PSP2 - learn defect and quality management
- PSP3 - scale up PSP methods to larger projects

The processes are said to be “defined” since that include precise and unambiguous procedures for carrying out the process. Each process is structured into a set of processes phases; each phase has a script that gives a step-by-step description of the tasks to be completed; and there are forms and documented standards that are used in carrying out the process tasks. There is also additional guidance and advice on how to analyse and improve one's process.

Reporting of actual industrial use data about the effects of the PSP has been limited by several factors:

- Many companies are reluctant to release specific data that competitors or customers could use to identify actual cost and defect levels.
- Most companies have little or no historical data to compare PSP data against, so it is difficult to quantify the effects that PSP had on their costs and schedule.
- PSP has not been widely adopted, so there are fewer cases available to draw from.

One of the first organisations to report their results after using both PSP and TSP was Teradyne. Their return on investment (ROI) analysis indicates that by using PSP and TSP on 2 projects totalling 112 KLOC, they have benefited by approximately \$5.3 million to date in saved engineering time. With code developed using TSP/PSP, they are saving approximately 120 hours/KLOC in integration, system, and field testing. Quality levels improved 20 times over prior projects and actual effort and schedule were within 8% of plan (early). Below are specific details about one of Teradyne's projects. Teradyne estimates the cost of PSP training to be one month per engineer. Boeing, AIS, and Hill Air Force Base have also reported receiving significant benefits from PSP and TSP use [3]. In classroom setting developers reduce the number of defects they leave in their programs while not changing their productivity [4]. In industrial setting estimate accuracy improves and the same happens to the quality of the product [3,5].

However, problems are reported too. El Emam reports a high rate of recidivism where following PSP training many workers do not persist with the disciplines and return to their original pre-PSP development practices [6]. Morisio, reports that the duration of training is often unsustainable for SME's, tool support is essential, human factors (such as privacy of data) are key, and finally that the PSP cannot be applied to individuals in isolation when they work in cohesive groups [7].

Feedback from PSP training programmes in Ireland has suggested that the absence of a support tool, to simplify the recording and analysis of the data produced is one of the major barriers to continued usage of the methods [8]. Developers tire of recording data on paper forms and eventually usage of the disciplines peters out. There is also corroborating evidence from the USA. For example, [9] and [10] report on experiences of a two year PSP study which questioned the quality of the data recorded. They found that there was significant data quality issues with manual PSP, for example, not all defects were recorded because the overhead in recording was too expensive.

So, although the ideas and disciplines associated with software engineering at the individual level found expression through the PSP, it was evident from the literature findings and the direct experience of the IPSSI consortium that a number of modifications to these approaches were required in order to gain industrial acceptance [11]. This included modifications to:

- Reduce the duration of training to make it more affordable.
- Tailor the ideas and disciplines for an industrial audience.

- Provide tool support for collection of data, computation and analysis of measures.
- Address human factors, build data privacy in the supporting tools.
- Support the transition phase from training to day to day usage in industry.

The following section describes a European-developed individual SPI framework “PIPSI” (Process for Improving Programming Skills in Industry), which was developed based on the above premise.

PIPSI

The IPSSI (Improving Professional Software Skills in Europe) project is an ESSI funded project which aims to provide a process improvement framework for use by individual software engineers working in European SMEs [11]. The focus of the project is on improving individual software engineering skills thus generating bottom-up improvement. At the heart of the IPSSI initiative is the PIPSI (Process for Improving Programming Skills in Industry) approach, whose aim is to present the techniques in a way that makes them more attractive and more easily used in small and medium-sized organisations and development teams. The projects three main deliverables are

- A personal process improvement methodology
- A set of configurable training materials
- A Personal project tool to support data gathering

The focus of the PIPSI is on bottom-up process improvement as illustrated in figure 2, which shows the three main elements of PIPSI personal software engineering:

- defining a personal process
- personal project management
- personal quality management.

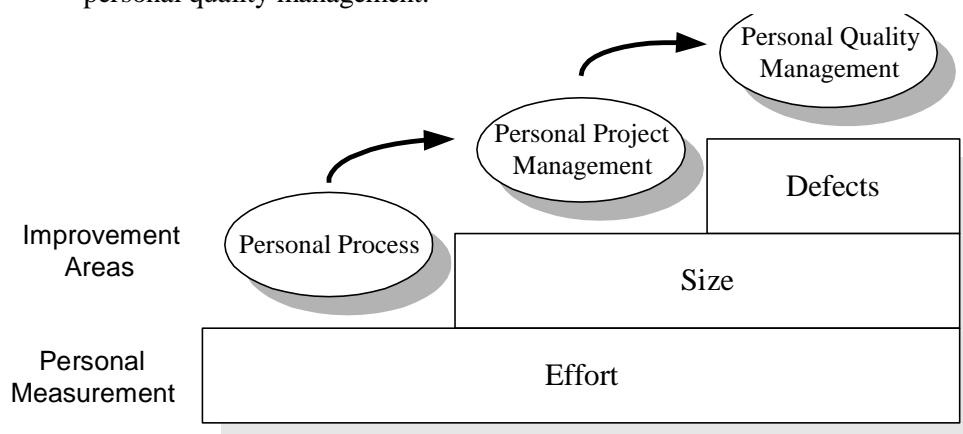


Figure 2 - The PIPSI Structure

The entire model is buttressed and controlled through the use of measurement. By collecting data on their own performance, software engineers learn about how they develop software. The measures help them understand the fundamental relationship between size and effort and, through this understanding, enable them to improve their estimating abilities. Furthermore, by gathering data on their defect rates they witness how employing practices such as personal code reviews and the use of checklists will allow them to produce higher-quality software products. The measures provide information on performance, information can then lead to process improvement and process improvement can lead to the production of better quality software on time. Finally collecting performance data on an ongoing basis moves developers from defining their own development process, through managing it to optimising it.

Through PIPSI training, developers complete programming tasks on which they collect increasing quantities of data. Early exercises capture effort measures. Subsequent exercises gather size data whilst the concluding exercises capture defect and quality measures. This is hugely empowering for both the programmer and the organisation as a whole. Programmers are now in a position where they can provide the project manager with achievable deadlines and the project manager can develop more accurate and predictable delivery schedules. The final element of PIPSI is that of personal quality management. As developers complete PIPSI program exercises, they collect data on the defects injected into those programs.

This process illustrates in which development phases they inject and remove defects. Furthermore, the defects are categorised by type thus allowing a causal analysis to be performed which can then lead to defect prevention. PIPSI focuses on proven quality control mechanisms such as design and code reviews which enable developers to remove defects earlier in the development process. This achieves the twin objectives of removing defects at the front end of the development cycle where they are cheaper and easier to fix and, as a corollary, means testing time is more focused as fewer defects are escaping into test.

As part of the PIPSI project, a tool set, which consists of data gathering and data analysis tools for use in a web-based environment, has been developed. The PIPSI support tool enables the measures to be collected as a simple complement to the development process. The tool also analyses the data collected to provide the developer with important process feedback.

Table 1 provides a summary of some of the key benefits of selecting PIPSI over PSP:

Table 1 – PIPSI Vs PSP

Selection Criteria	PSP	PIPSI
Developed for Industry	No – originally developed in academic context	Yes – developed specifically for SME's
Bureaucratic overhead	Very high	Reasonably low
Training time	2 Weeks	2-3 days
Cost of training courses	Very high	Moderate
Cost of training materials	Available only with training courses	Freely available
Tool support for data gathering	No – pen and paper or Excel spreadsheets	Yes – TIPSII tool designed specifically for PIPSI

Promoting PIPSI in Europe

The PSP is managed by the Software Engineering Institute (SEI) at Carnegie Mellon University, with training courses being conducted by both the SEI and their “Transition Partners” - authorised training centres - of which there are currently 11 in USA. In Europe today, there is a substantially less promotion of PSP, with organisations such as the European Software Institute (ESI) no longer offering public PSP training courses. In contrast to this, the IPSSI consortium have been actively promoting PIPSI across Europe.

During the early stages of the IPSSI project (1998-1999) the consortium partners conducted a series of awareness activities across a number of Europe countries (specifically, Ireland, Italy, Spain, France, Sweden, and Denmark). Typically they consisted of seminars or contributions to seminars, addresses to professional and industry associations, and short courses for managers. A technical briefing on PIPSI was developed, and used in mail-outs to promote PIPSI training. As well as promoting awareness of PIPSI in industry, some attention has been given to the academic community, with a series of talks and papers presented to groups interested in software engineering and process improvement throughout Europe.

At the start of the IPSSI project, a series of surveys was conducted in partners countries to assess both the level of awareness and usage of PSP (and the general level of SPI at the individual level). From the responses the following picture emerged:

- Ireland - as a result of training run by the Centre for Software Engineering, a number of individuals had been trained, but only one company have adopted PSP as its company process [8].
- Spain - although the ESI is a licensed PSP trainer, no Spanish respondents

reported using the PSP.

- Italy - one Italian company has adopted some elements of the PSP, in a project partnered by the Politecnico di Torino [7].
- Sweden - no Swedish SME's, who responded to the survey, were using the PSP.

PIPSI was marketed as a public course in Ireland, Italy, Spain and Sweden, and in France it was incorporated into industrial training, delivered as part of their normal business, by AFTI. The most successful take-up was in Spain, where a full public course with participants from industry was run. In Sweden, the course was run in-house in a software company in Gothenburg. In Italy and Ireland, the response from industry was insufficient to make a public course viable. In each of these countries the course was run for third-level students of software engineering. In Spain PIPSI training was taken up by practitioners from several different software companies, representing the industry, banking and manufacturing sectors.

The success in marketing the course in Spain is the more marked because previous attempts to run PSP courses there have failed, despite the ESI being an accredited PSP trainer. In contrast, the take-up in Ireland and Italy was low, although in both countries PSP courses have been successfully run. While there is interest in PIPSI in both countries, the consortium felt that the indications are that the perceptions of the PSP as being rigorous and expensive have carried over onto PIPSI, and made it more difficult to sell.

Also, in an Irish context, because of the relatively high awareness of PSP, organisations had already made a decision as to whether individual software processes were of any value to them. As such, despite the focused nature of PIPSI, organisations had already made their decision in this regard. In Ireland much of the interest in PIPSI was from organisations previously unaware of PSP or those who had not pursued training or time investment in this regard.

Throughout the project much interest was expressed in PIPSI by project managers as witnessed by the success in staging the PIPSI for Managers course which culminated in a successful tutorial presentation at EuroSPI 2000 [12]. However, whilst very supportive of the ideas contained within PIPSI managers decided against investing in practitioner training. Qualitative feedback and empirical comment in this regard, suggested that they were more likely to invest in what they perceived as skill-base enhancement such as training in new development languages or tools.

In the very recent past however, several companies have expressed an interest in PIPSI and have availed of presentations in this regard. As these companies are currently making process and training decisions, it is possible that PIPSI will be introduced into an industrial site in the near future.

The incubation time for technology adoption should also be borne in mind. For example it is almost 10 years since CMM was launched and 5 years since PSP was released, however it is only in recent times that these technologies are achieving widespread awareness. In marketing terms CMM, perhaps, has reached the "Early Majority" phase

whilst PSP is at the “Early Adopters” stage. PIPSI, on the other hand, is at the “Innovators” phase. It remains to be seen what will happen when PIPSI is used further in an industrial context as to whether this will make it sufficiently marketable to move to greater usage and awareness like its corresponding process counterparts.

PIPSI Case Studies

This section will provide an overview of the results of a series of selected case studies [13] based on PIPSI training as follows:

- Ireland – undergraduate students, Dundalk Institute of Technology
- Italy – undergraduate students, Politecnico di Torino
- Spain – industrial training, European Software Institute
- Sweden – postgraduate students, Chalmers University

In all cases PIPSI was tailored to the specific needs of the group being trained. However, all groups completed a series of 5 PIPSI exercises. During each of the 5 programming exercises participants are required to collect successively more detailed data. By following this approach, participants adhere to the PIPSI model by commencing with effort measures, then progressing to personal project management by relating effort to task size and finally focusing on quality management through understanding defects.

From the point of view of the PIPSI instructors, PIPSI was deemed very useful to give to the students an introduction to software engineering concepts and methods. The course was especially suitable for students unfamiliar with the detailed rigours of software engineering. The structure of the course, with theory and exercises to apply it in practice, was deemed very useful.

From the point of view of the students, the main comments were:

- Interest for the exposition to new concepts
- Appreciation for the practical application of concepts, with a nearly one to one correspondence between theory and practice
- Difficulty in agreeing on the interest of estimation. This depends both on the small time required for exercises, and in the lack of industrial experience. In fact, students with some form of experience of work in real settings were more interested in estimation.
- Difficulty in agreeing on the lifecycle and the three phases (pre build, build, post build). Again, this was due to the small time required by exercises. However, the students agreed that the problem was a problem of scale, and not conceptual.

Overall PIPSI training was found to be very effective in introducing the concepts of Software Process Improvement, and in making plain the benefits of the personal SPI techniques. The student responses to the course also indicated that they understood the need to collect metrics on their work.

Whilst the data collected does not provide conclusive proof of the benefits of using PIPSI there are many encouraging signs. Though time estimation has not improved during this instance of PIPSI training the practice of estimating time does possess particular difficulties. For example, the short exercises can generate quite large estimating errors, as participants tend to overcompensate for previous errors. Also, small tasks will always produce significant percentage errors, e.g. a 2-minute underestimate in a 20-minute task is a 10% error. It is expected that when applied to larger tasks and when size and productivity data are introduced that time estimates will be reduced.

The data does however, show improvements in size estimating and particularly in defect management. Again, the number of data points is insufficient to supply any definitive proof but the signs are encouraging. Indeed, the data should act as a momentum to participants to continue to use and monitor the PIPSI approaches to determine how they work for them in the longer-term.

There were also some qualitative benefits from the study. Many of the students commented on how they had not thought about programming in this way. A number expressed how previously, they were unaware of the proportions of time they were spending in the various development phases and furthermore had not taken product size into account.

For those who followed them most assiduously, code reviews provided the major benefit and, after the course, several developed a working checklist of potential defect causes to assist with their reviews.

The benefits, to the developer, of using IPSSI are ultimately self-convincing. The results above require supplementary exercises to allow more data to be collected and subsequently a clearer pattern of the developer's process. As IPSSI is aimed at industrial practitioners, these exercises could be real world projects. Ultimately, it is through applying IPSSI disciplines in their own daily work that developers can become convinced of its benefits.

Improving the quality of software products is a very important goal for both companies and developers. The trials show that if all of the exercises as currently designed are not completed then there is very little data on defects and the potential advantages of code reviews. It may be necessary either, to increase the number of exercises or introduce code reviews into earlier exercises. Either way this would provide more defect data and more feedback on the application of the reviews.

At present, within the exercises participants are asked in the later exercises to estimate the time required to complete an exercise based on the their estimate of the size of the program and their historical productivity. Whilst this is useful in highlighting the relationship between size, productivity and effort, no allowance is made for the complexity of the task.

Conclusions

The experience of the IPSSI consortium in investigating industry requirements and

defining an appropriate method has been worthwhile and rewarding. While the response from industry to the method developed has been slower than anticipated, the reaction has been positive, and it seems clear that there is a potential market for such methods. For example, there has been considerable interest from managers and academics who have heard about these methods, but little practical follow up.

In view of this, the consortium has decided that the best future for methodologies such as PIPSI lies in making them more widely and easily available. We have therefore decided to make PIPSI available as 'freeware', accessible from the IPSSI web page [11]. In this way, it is anticipated that a wider recognition of the method will grow, and individuals and companies will be encouraged to experiment with it.

The Consortium also notes that the companies (such as AIS, Motorola, Teradyne) that have had most success with PSP have previously invested heavily in CMM or other corporate SPI initiatives. Our experiment leads us to believe that PIPSI has a brighter future as a complement to corporate process improvement rather than as a stand-alone methodology.

Based on results to date [14], using a defined, planned, and measured personal process appears to offer many benefits. Assuming that further experience confirms these early results, one would expect methodologies such as PSP and PIPSI to be widely adopted and used. However, from experiments reported in this paper and elsewhere, it is also clear that industrial introduction is not easy.

Some SPI researchers believe that the future of such individual-level SPI methodologies is not as an as-is process for industrial contexts [7]. Rather, researchers, can use it as-is in academic teaching, where they can change context variables. Engineers can use it as a process template to inspire team or group process improvement.

Presuming the demand for skilled software engineers continues to increase, the most appropriate way to introduce SPI may be through the educational system. Rather than having students first learn undisciplined practices and then unlearn them, we should introduce disciplined methods at the beginning of the curriculum.

One avenue which may offer possibilities to PIPSI, is as a complement to SPICE (ISO 15504), where following PIPSI disciplines will support accreditation efforts. As SPICE is designed to be more easily tailorable for SMEs in comparison to the bureaucratically-heavy CMM, there may be potential gains and acceptance in this approach. There are also indications that PIPSI may have benefits in an eXtreme Programming (XP) environment. In XP, incremental development, with significant user input, means that careful estimation is required if timeboxes are to be adhered to. Also through this incremental approach as each new user feature is to be added to previously developed functionality then high quality control is necessary.

On a positive note ESSIE's such as PIOJAVA [15] have demonstrated that acquiring new technological expertise and process improvement can go hand in hand, and indeed that the latter can boost the effectiveness of technology acquisition. This is still a message that needs to be put across to hard-pressed European software development organisations, difficult though it may be both to change attitudes in today's skill hungry and fast changing technology market, and to explain the need to change a sceptical,

questioning European culture.

References

- [1] W.Humphrey, "Introduction to the Team Software Process", Addison Wesley, 2000.
- [2] W.Humphrey, "A Discipline for Software Engineering", Addison Wesley, 1995.
- [3] W.Humphrey, "Using a Defined and Measured Personal Software Process", IEEE Software, May 1996.
- [4] W.Hayes and J.Over, "The Personal Software Process (PSP): An Empirical Study of the Impact of the PSP on Individual Engineers", Technical Report SEI-97-001, Software Engineering Institute, December 1997.
- [5] P.Ferguson, W.Humphrey, S.Khajenoori, S.Macke and A.Matvya, "Results of Applying the Personal Software Process in Industry", IEEE Computer, May 1997.
- [6] K.El Emam, B.Shostak, N.Madhavji, "Implementing Concepts from the Personal Software Process in an Industrial Setting", Proceedings of Software Process 986, IEEE Computer Society Press, Brighton, 1996.
- [7] M.Morisio, "Applying the PSP in Industry", IEEE Software, December 2000.
- [8] P.O'Beirne and J.Sanders, "Personal Software Process: Does the PSP Deliver its promise?", Proceedings of Inspire '97, Gothenburg, Sweden, 1997.
- [9] A.Disney and P.Johnson, "Investigating data quality problems in the PSP", Proceedings of the ACM SIGSOFT Sixth International Symposium on the Foundations of Software Engineering, Florida, USA, 1998.
- [10] C.Moore, "Personal Process Improvement for the Differently Disciplined", Proceedings of the International Conference on Software Engineering, Los Angeles, USA, 1999.
- [11] IPSSI (Improving Professional Software Skills in Industry), ESSI Project number 27453, <http://www.compapp.dcu.ie/ipssi/>
- [12] "PIPSI", Tutorial presented at EuroSPI'2000, Copenhagen, Denmark , November 2000.
- [13] "Improving Professional Software Skills in Industry - A Training Experiment", R.O'Connor, H.Duncan, G.Coleman, M.Morisio, C.McGowan, C.Mercier & Y.Wang, Technical Report CA-0201, Dublin City University, 2001.
- [14] W.Humphrey, "The Personal Software Process: Status and Trends", IEEE Software, December 2000.

- [15] “Does Europe have the right attitude to process improvement?”, C.Chappell, Software in Focus, Issue 14, October 2000.

About the Authors

Rory O'Connor is a lecturer in Software Engineering at Dublin City University. He received a PhD. in Computer Science from The City University (London) and an M.Sc. in Computer Applications from Dublin City University. His research interests are centered on the processes whereby software intensive systems are designed, implemented and managed, in particular methods and techniques for supporting the work of software project managers and software developers in relation to software process improvement, software project planning and management of software development projects. He can be contacted by email: roconnor@compapp.dcu.ie

Gerry Coleman currently lectures in the Computing department at Dundalk Institute of Technology where he specialises in teaching Software Engineering and Project Management. He is currently engaged in research into software processes for small companies and in software failure analysis. Prior to joining Dundalk Institute of Technology, he was a Senior Consultant in the Centre for Software Engineering with responsibility for research, and technology training and transfer, in the areas of software process improvement, software metrics and software quality. He has previously worked for a number of years in the software development departments of several financial institutions as a Software Developer, Systems/Business Analyst and latterly as a Project Manager. He can be contacted by email: gerry.coleman@dkit.ie

Web based SPI – Benchmarking of Software – Organisations

Burkhard Neuper

Austrian Research Centers Seibersdorf, A – 2444 Seibersdorf

Manuela Stimpfl

Austrian Research Centers Seibersdorf, A – 2444 Seibersdorf

Introduction

The goal of everyone's business - including software industry - is to manage the 'magic triangle' of costs, time (to market) and the assurance of quality. The facts are well known that "Software is (almost) later than expected, more expensive than planned, and with less functionality than hoped [1]." The SPIRE Project in 1998 showed successfully that SMEs benefit from SPI using self-assessment tools like SynQuest or SPICE 1-2-1. Encouraged by this experience and the increasing demand of benchmarking data the Austrian Research Centers will introduce an international available SPI-benchmarking service in fall 2001. Until now the server is operating in test-mode and will go public in September 2001. The benchmarking server is a web based application administering currently more than 300 anonymous assessment data of former SPI Projects. The concept offers analysing, benchmarking and reporting services for individuals, consultants and companies who want to learn more about their organisations fitness. The paper will give a description about the basics of SynQuest and how the benchmarking services work.

SPI - Assessments using SynQuest and SPICE 1-2-1

The Process Improvement Center Seibersdorf (PICS) – a subgroup of the IT-Department for Safety and Security of the Austrian Research Centers Seibersdorf – is traditionally involved in matters of Software Process Improvement. PICS is a platform for SPI and Process Improvement in general supporting local Austrian consultants and their customers, especially SMEs.

The SPIRE Project

Due to the work of pioneers like Watts Humphrey (e.g. introducing CMM [2]) we know about the weaknesses we should avoid while developing software. From 1997 to 1998 the Austrian Research Centers Seibersdorf (ARCS) took part in the SPIRE Project among Partners of Sweden, Italy, Ireland and Great Britain to introduce Software Process Improvement practices. During the project 65 European companies have been coached in SPI and published via case studies. ARCS was responsible for the Austrian software industry, which is mainly represented by small and medium enterprises. A short survey in the year 2000 showed that SPI is still worshiped as a great benefit in 90% of the companies. The activities included so called ‘guided selfassessments’ which means that the company’s staff (CEO, project managers and software engineers) filled out together an electronic questionnaire. A coach assisted during the assessment which typically lasted about 5 –6 hours. After the assessments the companies got a complete feedback about their strengths and weaknesses. This procedure became known as the SPIRE – Method. [3]

The SynQuest - Tool

The electronic questionnaire used in Austria was SynQuest of HM&S Graz, Austria. A screenshot is shown in Fig. MSBN.1. The tool combines demands, hints and recommendations of Best Practices of software engineering including ISO Standards (ISO/IEC 15504 Software Process Improvement and Capability dEtermination SPICE[4] ISO/IEC 9000, ISO/IEC 12207, ISO/IEC 9004-4, ISO/IEC 9126), IEEE Documents (Software Engineering Project Management Glossary; IEEE 1058; IEEE Std 610.12), ESA PSS.05, CMM and from some experts (R.H. Thaye; Koontz, O’Donnell and Weirich; and Shooman).

As it takes 4 to 6 hours for one assessment it is very time- and cost – saving which meets one of the main demands of companies struggling with quality management tasks. The SynQuest assessment creates (controlled) time and (closed) space to discuss all the things which usually are just said during the coffee-breaks. After the assessment we often observe an effect of pure motivation because now the people who took part in the assessment want to realise their ideas of quality management and process improvement.. For many companies this is another important advantage of using SynQuest:

SynQuest is designed as a universal questionnaire, so it is easily adaptable to other topics too (e.g.: SW-Procurement: PROBE [5], SPICE 4 SPACE at ESA, CMM, EFQM [European Foundation of Quality Management], Tourism, Health, Ecological Systems, Controlling, Business, Quality-Management based on ISO

9000:2000,...). SPICE 1-2-1 is the big brother of SynQuest who strictly follows the guidelines of the ISO 15504. It is much more detailed, but it takes 2 to 3 times longer to work with.

SynQuest Charts

The output of the assessments are diagrams which show the organisations strengths and weaknesses. Some of the diagrams show the situation in detail like ‘All answers and their score’ or the ‘Ranking of answers’. Others are aggregating the results to groups as shown in Fig. MSBN.2 ‘Score Process Areas’ or ‘Score Process Attributs’ as Fig. MSBN.3 shows. ‘SPICE like charts’ show the 38 Processes of the SynQuest questionnaire using NPLF (N:‘not achieved’, P:‘partially achieved’, L:‘largely achieved’, F:‘fully achieved’) - and Capability Level (CL) – charts. Fig. MSBN.4 shows an examples for a NPLF-chart and Fig. MSBN.5. one for a CL-frequency distribution chart.

SPICE 98 mapped charts transform the results into ‘Real’ SPICE results, meaning that the 38 SynQuest Processes are mapped by calculation onto the 40 Processes defined in the SPICE Reference Model. This calculation is based on an algorithm developed by HM&S. It allows to compare results of SynQuest assessments with results of SPICE 1-2-1 assessments. Using this strategy the dataset of a SynQuest benchmark server may be extended by a subset of SPICE 1-2-1 assessments or vice versa. For benchmarking purposes this is an important feature for increasing and refreshing the actual datasets. For sure it is important to label original and calculated datasets to ensure the quality of the provided data.

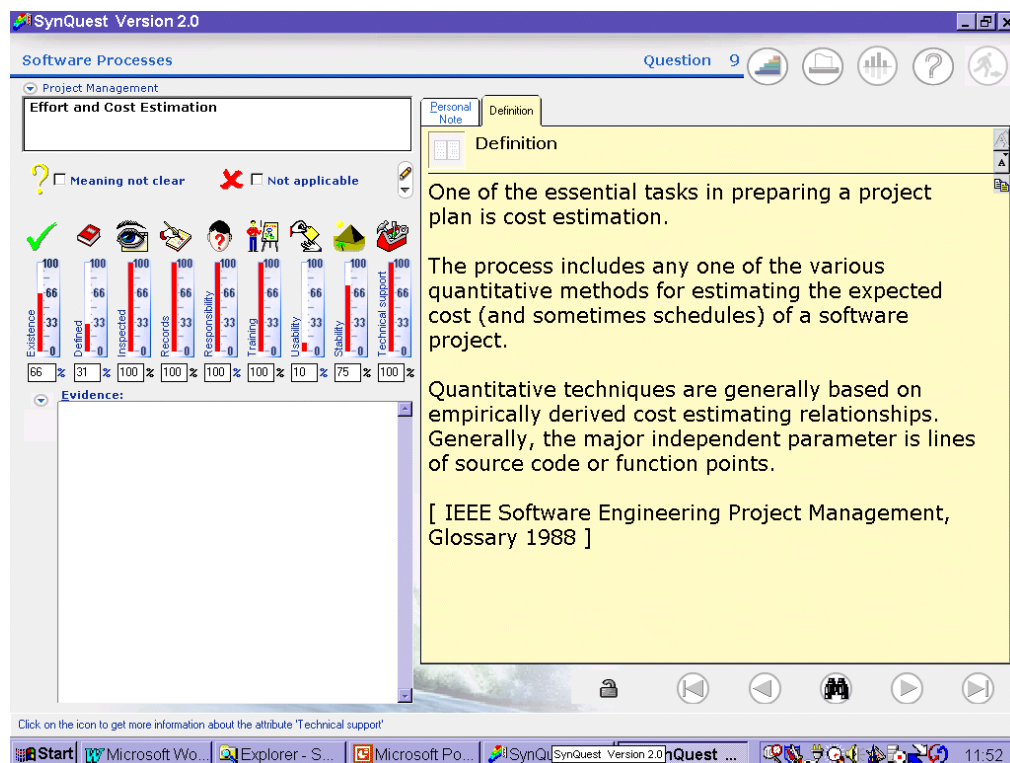


Fig. MSBN.1 SynQuest Screenshot

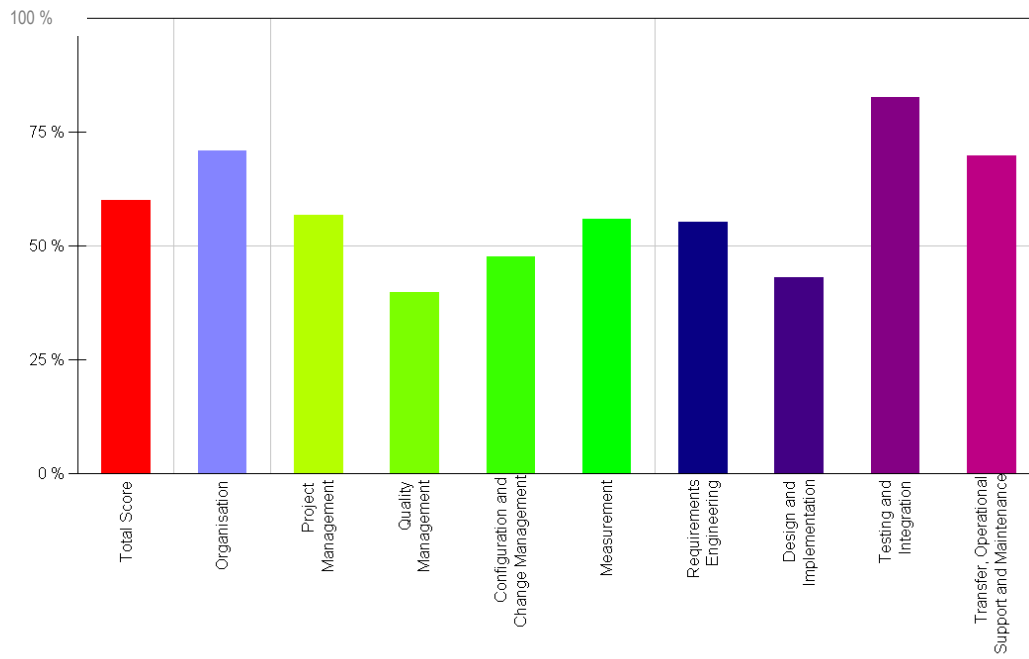


Fig.MSBN.2 Score Process Areas

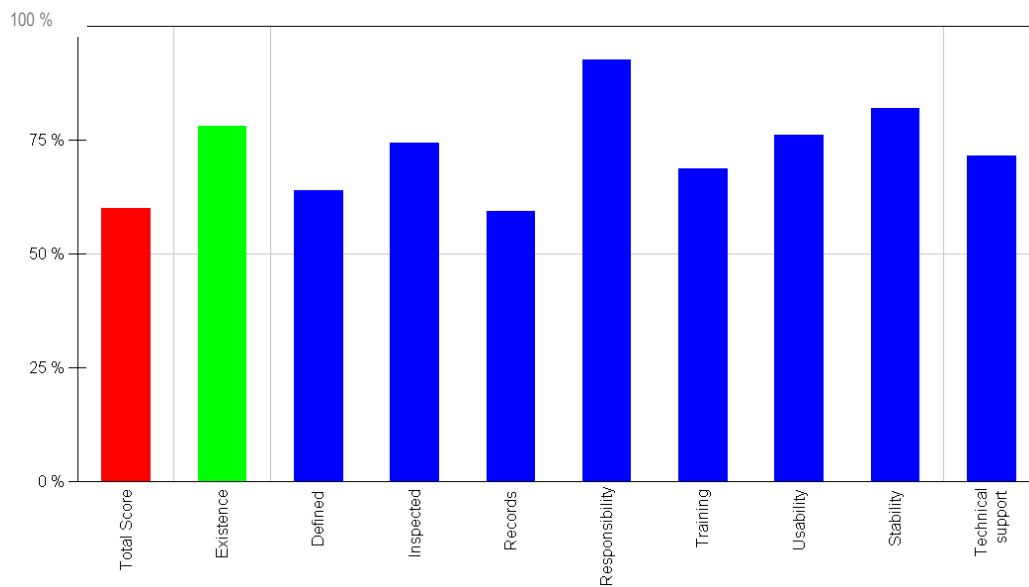


Fig. MSBN.3 Score Process Attributes

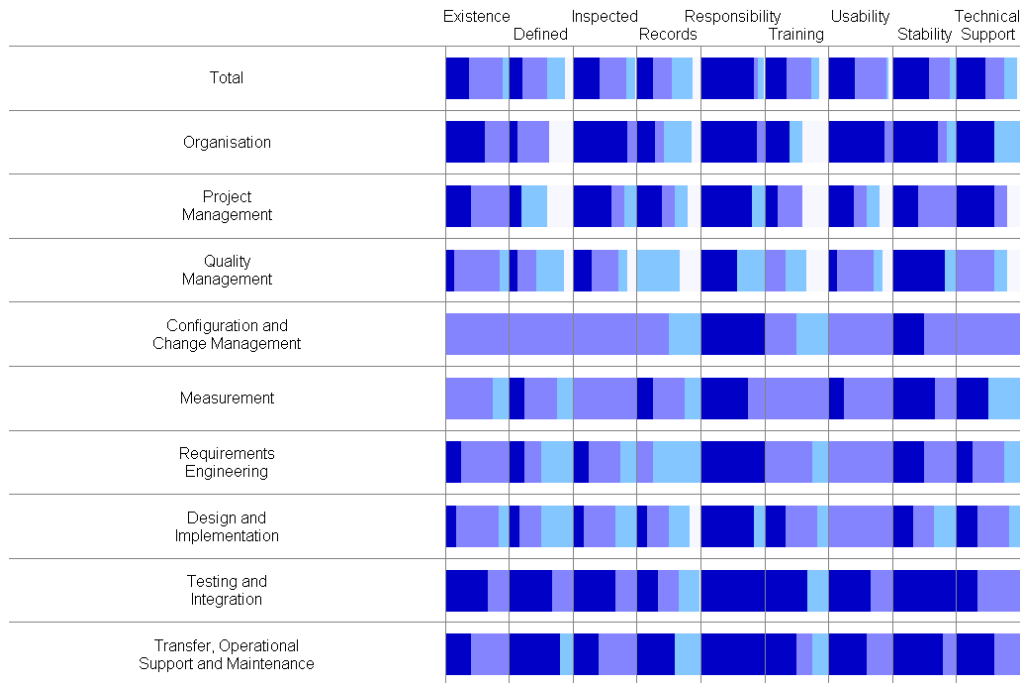


Fig. MSBN.4 NPLF - Chart

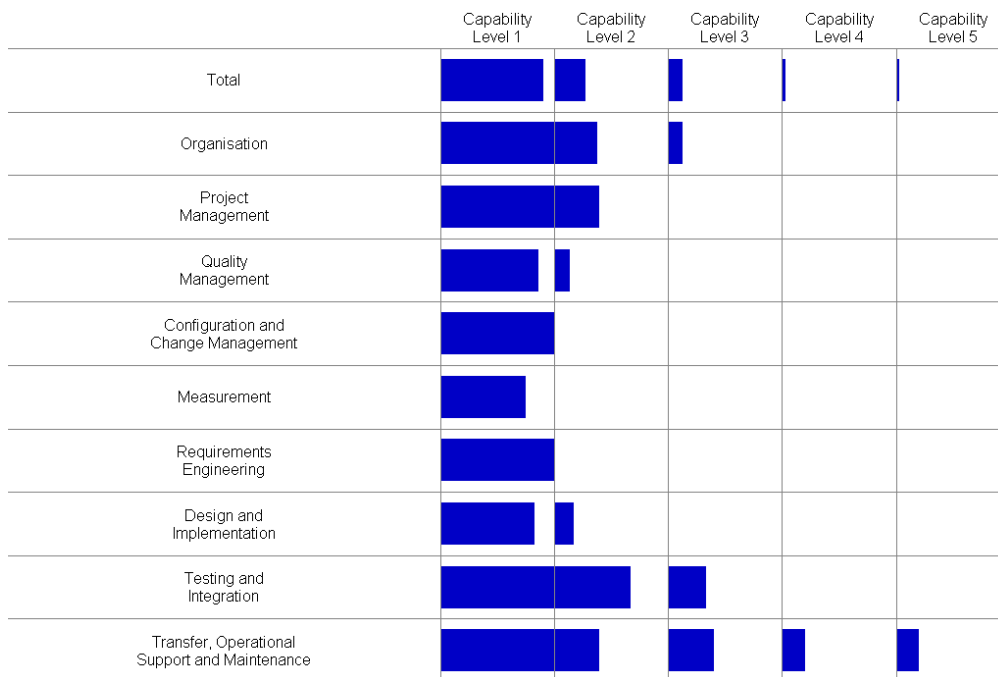


Fig. MSBN.5 CL - Chart

To ensure the comparability of the assessment data the 'General Questions' at the end of the assessment contain very important information. In these sections the organizations are asked about data about the company (e.g. name, department, address, year of foundation), employees (legal status, education, function, experience); portfolio (application domain, applied technologies); projects (employees involved, duration until delivery, number of projects currently in work, cost surplus); type of the organizations quality management initiative and about who filled in the questionnaire. In the benchmarking process this data is anonymized during the upload procedure described below. Finally the General Questions are closed by two feedback sections.

After an assessment the organization knows about its strengths and weaknesses. On which improvements they put the most emphasize often depends on the industry and environment in which they compete. As a result it will be necessary for many companies to know how they perform in comparison to other companies in their industry. Competitive advantage often depends on knowing the gap and as a consequence to close it. The benchmarking concept could be a possible method to do so.

SPI - Benchmarking

Process Benchmarking in general

The American Productivity & Quality Center (APQC) - the first International Benchmarking Clearinghouse - defines benchmarking as the process of identifying, understanding, and adapting outstanding practices from organizations anywhere in the world to help your organization improve its performance. Benchmarking should not be considered a one-off exercise.[6] To be effective, it must become an integral part of an ongoing improvement process.[7] At the EFQM (European Foundation for Quality Management) Excellence Model benchmarking is an integral part of the companies progress towards excellence.[8] The "classical benchmarking" approach requires to choose optimal benchmarking partners who are willing to exchange benchmark data. In addition to that the organization needs a deep understanding of the process being analyzed and the benchmarking process itself. [6]

The SynQuest – Benchmarking Process

The SynQuest methodology with standardized process areas and categorized answers allows to do benchmarking easily, anonymously and at lower cost. A company has to undergo an assessment with the SynQuest methodology and tools, transfer their anonymous data, specify the Benchmark areas they want to know and receive automatically a benchmarking report (see fig. MSBN.5).

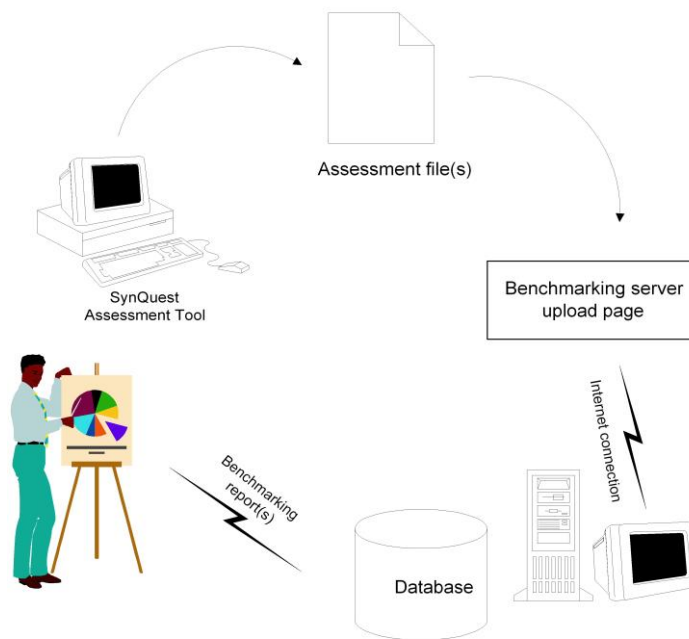


Fig. MSBN.5: Benchmarking with the SynQuest methodology

This procedure is realized by an internet based benchmark server – follow the link to <http://pics.arcs.ac.at>. The data is anonymized and integrated into the database by calculation during the upload to the server, so it is not possible to find out the companies identification. The Process Improvement Center Seibersdorf (PICS) has been established a benchmarking service and administers currently more than 300 anonymous assessment data based on SynQuest and SPICE 1-2-1 questionnaires. The benchmarking database grows with every submitted assessment result.

The next figure (fig. MSBN.6) shows an example evaluation of the Total Quality Quotient (TQQ) in different countries. The Total Quality Quotient measures the overall performance of an organization, concerning the processes organization, project management and quality management. The actual used assessment data base consists of 202 Austrian, 41 German, 45 Swiss and 7 assessments from other European countries. More than 70.000 different evaluations are possible, the next two figures only show examples of such benchmarking figures. Fig. MSBN.6 analyses the Total Quality Quotient in different countries. Another evaluation example could be the analysis of the Total Quality Quotient (TQQ) depending on the number of employees of the assessed company (fig. MSBN.7).

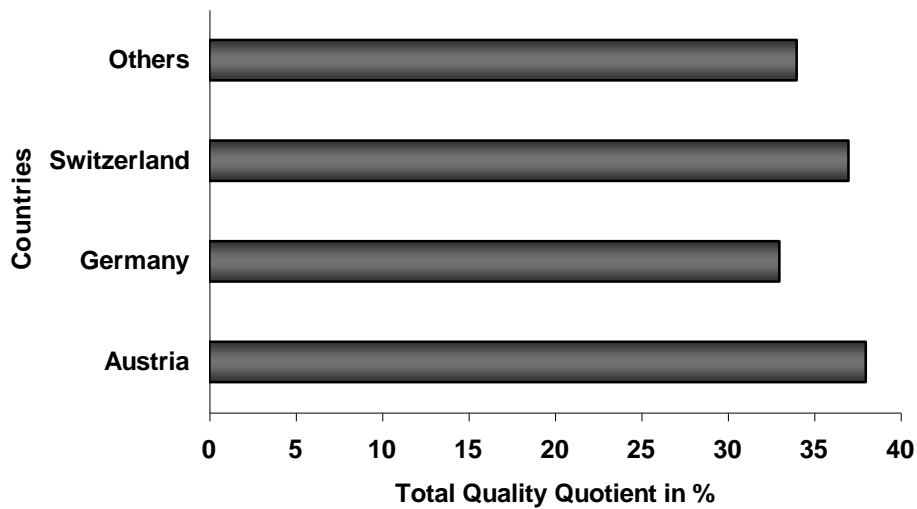


Fig. MSBN.6: Total Quality Quotient (TQQ) in different countries

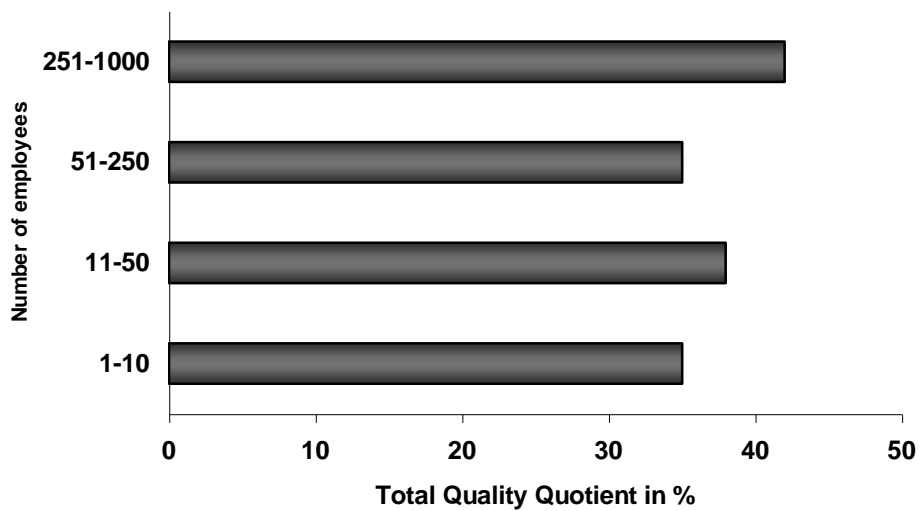


Fig. MSBN.7: Total Quality Quotient (TQQ) versus company size

The next figure (fig. MSBN.8) shows the Total Quality Quotient (TQQ) of one example assessment in comparison with all selected other assessments. The chart shows that the assessment has a better Total Quality Quotient than 80 other assessments, but 220 assessments have reached a higher value (and this case are 'better') in the term of Total Quality Quotient.

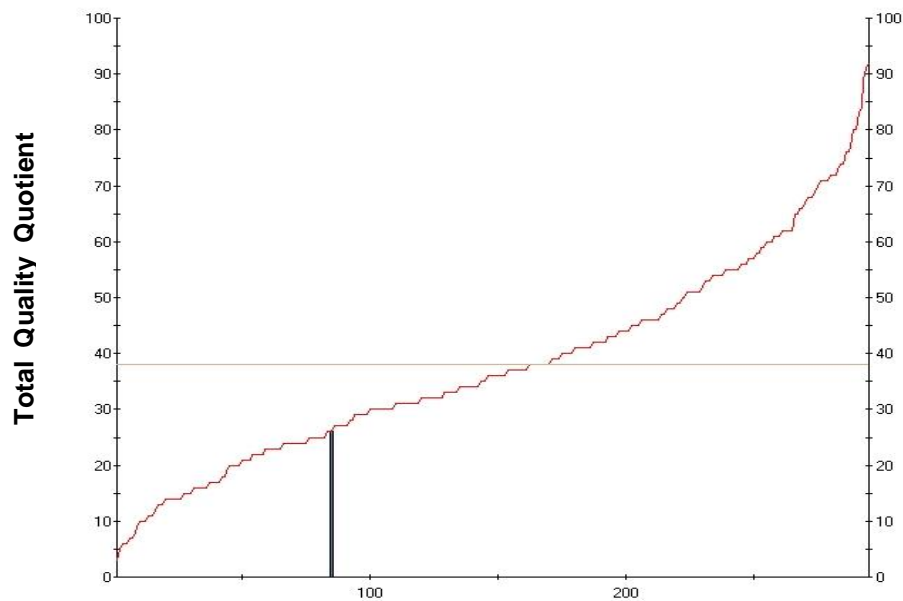


Fig.
MSBN.8.: Total Quality Quotient (TQQ) in order of all assessments

Implemented Benchmarking Services

The benchmarking server is a web based application (<http://pics.arcs.ac.at>), which permits analysing, benchmarking and reporting of data generated by the *Self-Assessment-Tools SynQuest* and *SPICE 1-2-1*. Registered users at the Process Improvement Center Seibersdorf are able to upload their *assessment data* over the Internet and compare them with anonymized *assessments* stored at the benchmarking server.

A user of the benchmarking service can generate a *report* on the fly or can subscribe such a service. The target group of such a service are

- organizations who would like to improve their processes. There are specific benchmarking solutions possible, like a benchmarking server for one company.
- or consultants in the area of (Software) Process Improvement and Benchmarking.

Until now the following services are implemented:

Basic:

Organisations may benchmark their data based on predefined standard criteria. Three different report types (standard, extended and advanced) are available which differ in number and depth of compared criteria.

Consulting:

A special service for consulting purposes. The data of several assessments can be managed, calculated and benchmarked using predefined subsets of criteria up to the full usage of the server. Individual definitions of compared criteria are necessary.

Company:

Large organisations usually want to manage their own data and like to benchmark their own divisions on company level as well. This service provides internal operation of benchmarking services without buying an own server. Individual definitions of compared criteria are necessary.

Conclusion

SPI and benchmarking are topics of interest. Especially for SMEs a concept of gathering and benchmarking data by using self-assessment tools like SynQuest and the according web-based benchmarking server should be of interest, as it can be done mostly by the organizations themselves. Therefore it is saving time and costs which is an important argument in performing SPI – or not - in SMEs, as former projects like SPIRE showed. The benchmarking server of Austrian Research Centers tries to meet this demands. Currently it is running in test-mode so there will be first practical experiences until the date of the conference.

Lessons learned

As the SPI-benchmark server is currently in the testing phase it is hard to talk about lessons learned now. Until the conference in October there should be some material about the first months online including topics about customer acceptance, operational- and performance behaviour.

The benchmarking services are presented at the EuroSPI conference to discuss the service during its very first phase in operation and to get some first feedback.

Appendix A: Authors short CV

Burkhard Neuper (burkhard.neuper@arcs.ac.at) received his degree in Telematics at the Technical University of Graz, Austria. He has been working for the Austrian Research Centers since 1999 in the Department of Information Technologies in the field of Process Improvement. As a member of the Process Improvement Center Seibersdorf (<http://pics.arcs.ac.at>) he is involved in Software Process Improvement, Benchmarking and Business Process Engineering. Main businesses are project management, consulting and product development for process improvement services.

Manuela Stimpfl (manuela.stimpfl@arcs.ac.at) received her MBA degree at the University of Klagenfurt. She has been working for the Austrian Research Centers since 2000 in the Information Technology Department focusing on Process Improvement. In the Process Improvement Center Seibersdorf (<http://pics.arcs.ac.at>) she works in the field of Software Process Improvement, Benchmarking, Business Process Engineering and Knowledge Management. The main businesses are project management, product development and communication activities for the process improvement services.

Appendix B: Short Company Profile

PICS is a service of the Austrian Research Centers Seibersdorf (ARCS).

ARCS is a research center for the private sector and government agencies with more than 700 employees at various locations across Austria. It is the largest application-oriented information enterprise in the country.

PICS works in the areas of

- Software Process Improvement (SPI)
- Business Process Engineering (BPE)
- Process Benchmarking

Our clients and the improvement of their business processes are in the center of our interest. We help our clients by analysing the business processes, recognizing unknown processes and proposing improvement measures

PICS is experienced in the areas of

- practical use of SPI and BPE (for example in the SPIRE Project – Software Process Improvement in the Regions of Europe)
- scientific occupation with SPI and Business Process Engineering
- continuous development of the methods

- experienced set up of networks and cooperation

We pursue a systematic approach to get a precise improvement with quick visible benefit. For this purpose we offer, SPI (self-)assessments and audits, consulting, benchmarking-services and assessment tools to gather quick and easy the necessary data.

References

- [1] Wallmüller E. (2000): Process-oriented Software Quality Management, Informatik, No. 3, 2000, p. 8-12
- [2] Humphrey, W.S.(1989): Managing the Software Process, Addison – Wesley Longman Inc.
- [3] The SPIRE Project Team(1998): The SPIRE Handbook: Better, Faster, Cheaper Software Development in Small Organisations, The European Commission
- [4] ISO/IEC 15504 –2 (1999): Information Technology – Software Process Assessment Part 2: A Reference Model for Processes and Process Capability, pp. 1-39
- [5] Wang, Y. (2000): A recent Extension of ISO 15504 to IT Acquisition Processes, Proceedings of the EuroSPI 2000 Conference, pp 1-2 ...1-13
- [6] American Productivity & Quality Center (Mai 2001): www.apqc.org
- [7] Sean O'Reagan, Richard Keegan (2000): Benchmarking Explained, Benchmarking in Europe, p. 10
- [8] European Foundation for Quality Management (June 2001): www.efqm.org

Additional Publications & Vendor Track

Session Chair:
Richard Messnarz, ISCN, Ireland

PICS (Process Improvement Centre Seibersdorf)	A-2
Requirements Based UML	A-9
ISCN (International Software Consulting Network)	A-23
EuroSPI Partners	A-26

Process Improvement Center Seibersdorf (PICS) – Products and Services

Burkhard Neuper

Austrian Research Centers Seibersdorf, A – 2444 Seibersdorf

burkhard.neuper@arcs.ac.at; <http://pics.arcs.ac.at>

Manuela Stimpfl

Austrian Research Centers Seibersdorf, A – 2444 Seibersdorf

manuela.stimpfl@arcs.ac.at; <http://pics.arcs.ac.at>

PICS Company Profile

PICS is a service of the Austrian Research Centers Seibersdorf (ARCS). ARCS is a research center for the private sector and government agencies with more than 700 employees at various locations across Austria. It is the largest application-oriented information enterprise in the country.

The Process Improvement Center Seibersdorf (PICS) – a subgroup of the IT-Department for Safety and Security of the Austrian Research Centers Seibersdorf– is traditionally involved in matters of Software Process Improvement. PICS is a platform for SPI and Process Improvement in general supporting local Austrian consultants and their customers, especially SMEs.

PICS works in the areas of

- Software Process Improvement (SPI)
- Business Process Engineering (BPE)
- Process Benchmarking

Our clients and the improvement of their business processes are in the centre of our interest. We help our clients by analysing the business processes, recognizing unknown processes and proposing improvement measures

PICS is experienced in the areas of

- practical use of SPI and BPE (for example in the SPIRE Project – Software Process Improvement in the Regions of Europe)
- scientific occupation with SPI and Business Process Engineering
- continuous development of the methods
- experienced set up of networks and cooperation

Products and services

We pursue a systematic approach to get a precise improvement with quick visible benefit. For this purpose we offer

- SPI benchmarking-services,
- assessment tools and
- consulting in the areas of SPI and BPE.

Benchmarking Services

The benchmarking server is a web based application (<http://pics.arcs.ac.at>), which permits analysing, benchmarking and reporting of data generated by the *Self-Assessment-Tools* SynQuest and SPICE 1-2-1. Registered users at the Process Improvement Center Seibersdorf are able to upload their *assessment data* over the Internet and compare them with anonymized *assessments* stored at the benchmarking server.

A user of the benchmarking service can generate a *report* on the fly or can subscribe such a service. The target group of such a service are

- organizations who would like to improve their processes. There are specific benchmarking solutions possible, like a benchmarking server for one company.
- or consultants in the area of (Software) Process Improvement and Benchmarking.

Until now the following services are implemented:

- **Basic:**

Organisations may benchmark their data based on predefined standard criteria. Three different report types (standard, extended and advanced) are available which differ in number and depth of compared criteria.

- **Consulting:**

A special service for consulting purposes. The data of several assessments can be managed, calculated and benchmarked using predefined subsets of criteria up to the full usage of the server. Individual definitions of compared criteria are necessary.

- **Company:**

Large organisations usually want to manage their own data and like to benchmark their own divisions on company level as well. This service provides internal operation of benchmarking services without buying an own server. Individual definitions of compared criteria are necessary.

Assessment tools

Assessment tools are electronic questionnaires for quick and easy data gathering. In cooperation with HM&S we are able to offer the following assessment tools:

SynQuest

Evaluation of the quality of your IT and software development activities: precise, comparable and efficient.

SPICE 1-2-1

Assess your IT processes compliant to ISO 15504 (SPICE)

Eco-Quest

Assess your process quality in the area of e-business

Assess&Act ISO 9000:2000

Assessment tool for business and management processes for all industries compliant to ISO 9000:2000.

My Business Quest

The management may assess the own business processes efficiently and less time consuming.

Consulting

After performing an assessment, the improvement potentials of your business processes obtained from the assessment results should be implemented in your business processes and lead to a continuous improvement process. On this occasion we offer our help to take the first steps and then to initiate a continuous improvement process.

We offer support in following areas:

- project planning and controlling
- continuous improvement process modeling
- tools and method selection
- improvement implementation
- issues regarding the quality management systems
- software validation and certification

Contact

Process Improvement Center Seibersdorf (PICS)

A-2444 Seibersdorf

Phone: +43(0)50550-3157

Fax: +43(0)50550-72133

e-mail: pics@arcs.ac.at

<http://pics.arcs.ac.at>

REQUIREMENTS-BASED UML

Joseph D. Schulz

Technical Director, International Channels

*Technology Builders, Inc.
400 Interstate North Parkway, Suite 1090
Atlanta, Georgia 30339
USA*

Fax: (815) 333-0160
Email: joes@tbi.com

Abstract

The purpose of this paper is to describe the "Requirements-Based UML" (RBU) development technique. RBU is a straightforward, *pragmatic methodology* for integrating structured requirements analysis into a UML-based analysis and design effort. It involves a very high degree of customer participation and involves the creation of measurable requirement definitions before each stage of modeling and/or coding. RBU includes only the essential tasks and is designed to be highly communicative and easily understood by both customers and professional development staff. Most often developed in direct cooperation with customers via a "Joint Application Design" (JAD) approach, the requirements are used to both design and validate the application functionality.

This paper only includes a brief description of the RBU process. Accordingly, it is not meant as a complete implementation guide for a professional development organization. RBU's major tasks and techniques are described here, but there has been no attempt to include all of the necessary components of a robust methodology (e.g., standards, procedures, forms, etc.). In addition, the examples contained within are merely illustrative of the overall approach.

Major Learning Points

After reviewing this paper, the reader should gain additional insight into the following areas:

- *The basic structure of the Unified Modeling Language (UML) development process*
- *The purpose and benefits of a structured Requirements Management (RM) process*
- *The RBU approach to incorporate structured RM into UML*
- *The appropriate "levels" of requirement information within the RBU context*
- *The importance of traceability across the development framework*

Keywords:

Requirements; Object-Oriented; JAD, UML; Quality Assurance; Application Development; Coding; Collaboration

Requirements-Based UML

Introduction

The Dreaded "M" Word

Every project needs one and every developer follows one, whether formal or informal, prescribed or ad-hoc. Unfortunately, for most IS professionals the word "*Methodology*" invokes dreadful images of the worst kind. It often implies reams of unnecessary work, impossibly rigid standards, and lots of wasted time. When the average developer hears the dreaded word, they usually assume that the related project is doomed.

Of course, just the opposite is true. A development project that doesn't actively use some sort of methodology has relatively little chance of success. If such a project does succeed, it is merely through coincidence or sheer dumb luck. These are the kinds of projects that ramble about generating lots of paperwork but relatively few measurable results. They miss every major deadline because they change directions so frequently, and usually require large quantities of rework to "fix" previous mistakes. In a project without a methodology, there is usually no such thing as a *frozen* deliverable, so consequently there are *no* deliverables.

So, then what exactly is a methodology? In its simplest form, a methodology is a set of steps to accomplish a task. That's it. No fancy buzzwords or expensive terminology, just a set of steps. It is a plan that describes each task and its sequence relative to the others. After all, any job worth doing is worth planning for. As the old military adage goes, "If you fail to plan, you are planning to fail."

Of course, a robust methodology can also include many other components. Strictly speaking, a complete methodology includes not only task descriptions but also supporting components like task standards, technique guidelines, deliverable outlines, and quality metrics. These items, though, are merely present to supplement the basic purpose of the methodology, which is to identify and prioritize the work to be done. That is, to describe the set of steps needed to accomplish the goal.

Unified Modeling Language

The latest emerging industry-standard in the object-oriented methodology arena is the *Unified Modeling Language*, commonly referred to as "UML". UML is a collaborative effort between the "Three Amigos" of the object-oriented analysis and design (OOAD) industry, i.e., Grady Booch, Ivar Jacobson, and Jim Rumbaugh. Each of these three had previously authored their own competing methodologies and realized the significant benefits of a truly global standard for (OOAD). By combining much of their previous work, the UML standard was born.

Of course, UML is not actually a methodology. Rather, it is a notational standard that can be used to implement the tasks within a methodology. By having a common notation, methodology and tool vendors can easily develop complementary solutions without requiring retraining of the workforce. UML-based methodologies define differing sets of tasks, but the techniques all employ the standard graphical symbologies.

One such example is the "Rational Unified Process" (RUP) from Rational Software. RUP is a complete methodology developed by the "Three Amigos" which uses the UML notation to represent all of its deliverables. It includes suggested task plans and also defines guidelines and metrics that can be used to manage and measure the development process.

Use Case Models

The UML specification includes graphical notations for many different diagram types, with most being optional steps based on the complexity of the application being developed. Relatively speaking, the "first" deliverable described in the UML notation is the Use Case Diagram. A Use Case Diagram graphically depicts the interaction between system users (i.e., "actors") and system functions (i.e., "use cases"). Subsequent UML diagrams build on this basic information to identify the system components, methods, and packages necessary.

Unfortunately, this focus on beginning with use cases creates a glaring deficiency in most UML-based methodologies, that is, they don't address the business-oriented application requirements. Instead of first defining the purpose and objectives for the development effort, UML methods begin by jumping directly to the software functions. This presupposes that the use case participants already know why they need a system and what the optimal solution should look like.

In reality, the most important part of any systems development effort is to first establish a firm understanding of the problem so that potential solutions can be effectively weighed. To do this, the key business requirements must be defined, including the return-on-investment justification for each. Once this *Objective Baseline* is established, proposed alternatives can then be measured to determine which best solves the stated problem. Without this requirements analysis, a UML-based approach may only help to deliver the *wrong* application faster and cheaper.

Requirements-Based UML

One possible solution to this problem is the use of "*Requirements-Based UML*" (RBU). RBU is a structured approach for incorporating business-oriented requirements analysis into a UML-centric development method. It balances the need for non-technical business analysis against the need for the structured technical approach defined in UML. Furthermore, it identifies business requirements analysis as a precursor to software-centric use case modeling efforts.

RBU also relies on a more natural, textual format for requirements deliverables. Non-technical staff members are generally more comfortable with words than diagrams, so RBU business requirements are defined in sentences and paragraphs. These textual descriptions are then related to the graphical objects defined in the UML deliverables.

After the first level of UML diagrams is completed (use case models, collaboration diagrams, etc.), the requirements are refined into more detailed textual technical specifications. In turn, these specifications are then related to the next round of UML diagram objects. This process of textual requirements leading UML modeling can continue to whatever level of detail is appropriate for the specific project.

By using this alternating approach with requirements and diagram objects, a more complete analysis and design model is produced. This provides a clearer picture of the application environment, including not only answering the "How?" questions for the application but also clarifying the "Why?" and "What?" as well. All too often development

teams are eager to rush into coding and the latter two questions remained unvisited. It is these types of projects that are most often cancelled or rejected by the customers because they provide little business value.

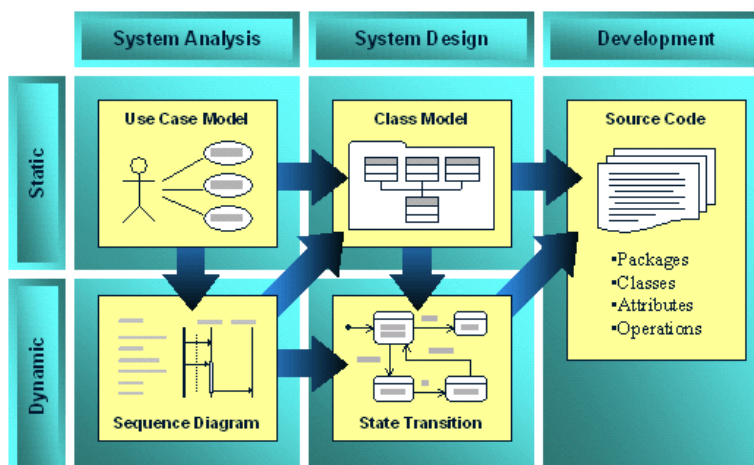
UML Overview

What is UML?

The Unified Modeling Process (UML) is a common notation for structured modeling within an Object-Oriented Analysis and Design (OOAD) framework. It was originally developed by several of the leading OOAD methodologists as a means to help standardize the types and format of deliverables produced by the competing OOAD methods. While not strictly a methodology itself, UML describes the notation that methodology outputs employ.

The current UML notational standard addresses the system analysis, design, and deployment steps in a development lifecycle. This version of the UML, v1.3, was approved in June 1999 by the Object Management Group (OMG). A new draft standard, v2.0, is currently in RFI review and will extend the current standard to include a range of other activities. The most notable addition expected in v2.0 is a common notation for business process redesign.

A Typical UML Process



A UML-based development methodology usually involves a series of graphical models which are used to define the functional and technical aspects of an application system. Each model depicts a diagrammatic representation of one aspect of the application and is integrated with the

other model objects. These models are then used as the component specifications for the construction phase of the project.

As shown in the graphic, the first and primary model developed in most UML-based methods is the Use Case diagram. Use Case diagrams are used to identify the external system boundary for an application by depicting the system functions ("use cases") that external entities ("actors") are able to interact with. Use Case diagrams are generally developed in very close collaboration with the application's ultimate customers or sponsors.

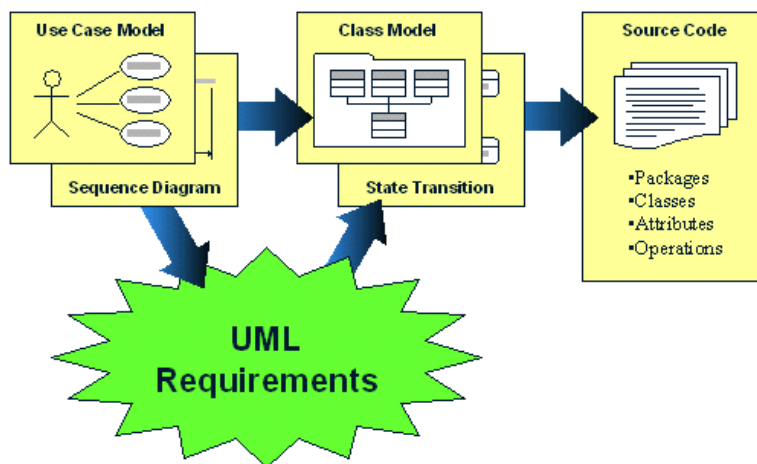
These Use Case diagrams are often then more fully described by the creation of either Object Sequence diagrams and/or Object Collaboration diagrams. Both of these diagram types serve to more fully describe the Use Case by including the nature and order of each of

the major work steps within the Use Case. Together, some combination of these three diagrams provide the functional application requirements for a system.

At the beginning of the system design, then, the system analysis models are used as input to create the relevant Class diagrams and/or State Transition diagrams. These design models describe the technical structure of the application. Any of several deployment models (e.g., Package Deployment diagrams, etc.) may also be defined in order to complete the technical specification before code development begins.

UML Requirements

In the context of a UML-based method as outlined above, the term "requirements" generally refers to a set of technical specifications that describe the software features in an application. These requirements are imperative statements of functionality that must exist in the developed code and are written as "Plain Language" textual sentences or paragraphs. This deliverable is often named the "System Requirement Specification" (SRS).



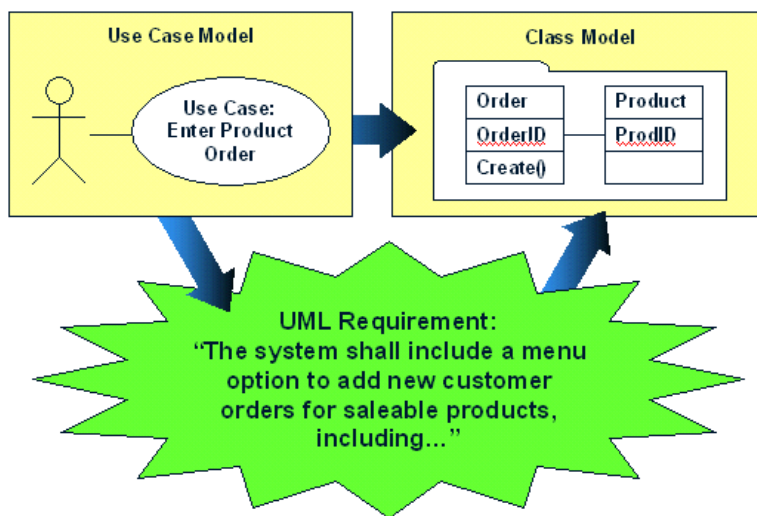
Most often, these "UML requirements" included in the SRS are developed as extensions of a Use Case diagram. For each Use Case defined, the complete set of mandatory characteristics is identified and documented in clear, concise

language. Modelers will then use these requirement definitions to help complete and validate the systems design models, ensuring coverage of all required functionality.

The SRS generally includes both functional and non-functional requirements. Functional requirements state a capability that invokes or performs an actor-oriented transaction. Non-functional requirements, on the other hand, state a characteristic of the application which limits or bound a designer's ability to develop a solution. Non-functional requirements usually include information about traits like performance and capacity limits, security rules and responsibilities, and/or technological considerations.

UML Requirements Example

In the example pictured below, a Use Case has been defined named "Enter Product Order". This Use Case would exist on one or more Use Case diagrams and would be detailed with the inclusion of a Use Case narrative (e.g., pre-conditions, post-conditions, etc.). In the diagram, the appropriate actor(s) would also be associated to the Use Case.



As part of the transition from system analysis to system design, UML requirements would then be defined for this Use Case. These requirements would itemize the specific features necessary in the software in order to fully accomplish the "Enter Product

Order" Use Case. As mentioned above, this might include both functional requirements and non-functional requirements.

One such UML requirement for the "Enter Product Order" Use Case might be the statement that "The system shall include a menu option to add new customer orders for saleable products...". As a result of this requirement, when the user interface class is developed in the Class diagram the system designer will know to include a method to invoke this process. This may also be reflected in the appropriate State Transition diagram(s) as an event which triggers a change in state.

Benefits of UML Requirements

The advantage of developing a structured SRS which includes both models and textual requirements is twofold. Firstly, the textual statements are often more communicative than UML notation to non-technical customers as they provide a written description of the functionality that anyone can read. The graphical notations can often be daunting for an uneducated customer, and so the textual descriptions are more comfortable for those without any previous UML training.

Secondly, the textual requirements provide a place to document software features that may not be readily apparent or do not exist in the graphical models. For example, non-functional characteristics like hardware constraints are difficult to include in UML models because they are typically global issues that cannot be incorporated into the description of just one model object.

So, UML requirements become the document-centric "bridge" between the graphical system analysis deliverables (Use Case diagrams, etc.) and the graphical system design deliverables (Class diagrams, etc.). Most often these requirements are managed with a word processing application and reviewed and approved in document format. This comfortable paradigm mimics traditional document-oriented analysis techniques.

Drawbacks to UML Requirements

However, there are also disadvantages to limiting the requirements process to technical feature descriptions. First and foremost, by beginning the analysis process with Use Case definitions, the focus is immediately on the design of the software. Since Use Cases describe

systemic solutions to problems, the derived UML requirements will address only the systemic characteristics as well.

With this approach, the only solution that can be developed will be one that can be automated with an application. This virtually ignores the relevant business issues that may be all or part of the problem as well. Often a minor business process redesign (like job function reorganization) can facilitate a more efficient application or even eliminate the need for an application at all.

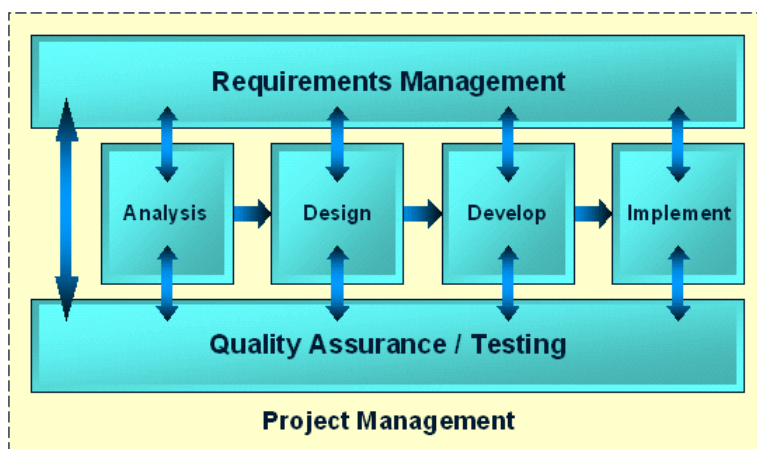
Also, UML requirements tend to include a lot of technical language since they are describing technical features. This is typically because they are written for the development organization to use as an input to the system design process. However, another goal of a structured requirements analysis is to validate the system analysis deliverables and the customers needed to do this are often non-technical. So, the personnel with the appropriate business knowledge may not be able to adequately understand the requirements definitions.

Finally, another problem with UML requirements is that they tend to focus on one business transaction at a time. Since they are most often derived from Use Cases, the requirements are documented with an eye toward that one transaction and often ignore the business workflow surrounding it. Without a highly structured reuse analysis, it is often possible to end up with highly efficient transactions that contain a lot of business redundancy between them.

Requirements-Based UML (RBU) Overview

What is RBU?

Requirement-Based UML (RBU) is a structured approach for integrating formal requirements analysis into a UML-based analysis and design effort. It balances the need for non-technical business analysis against the need for the system-oriented approach defined in UML by including a multi-level requirements definition. Instead of just the technical feature descriptions captured in traditional UML requirements (see previous chapter), RBU defines multiple requirements deliverables with a specific focus for each. Simply put, requirements management becomes a lifecycle task that runs in parallel with the OOAD tasks.



As with UML requirements, RBU requirements deliverables are defined in a natural, textual format. This allows non-technical customers to more comfortably review and understand the requirements information. These textual descriptions are then related to

the relevant graphical objects defined in the UML-based deliverables.

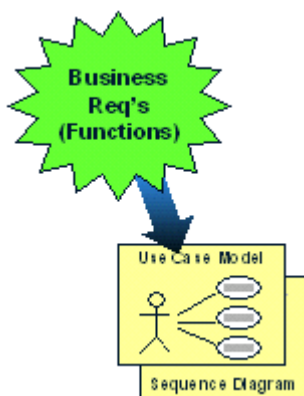
Often, development teams prematurely rush into the development of the application solution and ignore the larger business issues. This can easily lead to inappropriate or expensive technological solutions. By using alternating "rounds" of modeling and textual specifications, the RBU approach helps to temper that tendency and delivers a richer, more complete picture of the business problem. In this manner, it helps to ensure a higher quality, more cost-effective solution.

In addition, the RBU approach addresses Quality Assurance as a lifecycle task as well. Requirements are thoroughly tested before any development is performed, detecting conflicts and omissions that would stall later development. At each stage, the QA/Testing plan is refined and more detail is added until specific test cases have been identified. By developing the test cases from the requirements rather than the code, a more complete test harness is established.

Typical RBU Process

The RBU technique begins with a textual specification of all of the requirements for any solution to the business problem. These requirement statements define both the functionality required in the solution as well as the boundaries the solution must operate within. All of these requirement statements should be specified in a non-technical, "plain language" format. Ideally, the customers will define these textual requirements themselves without restatement by the development staff.

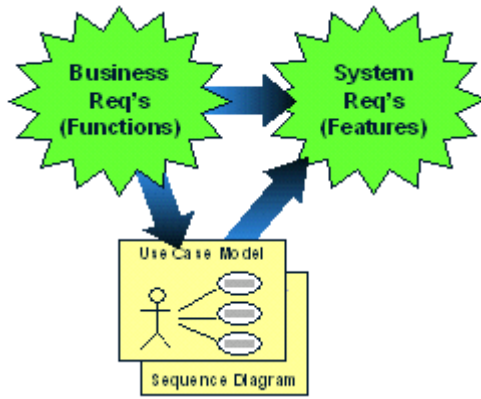
These requirements, usually referred to as "Business Requirements", should be defined without regard to how the application will look. Specifically, they should not reference menu options or screen formats. The intent is to capture a definition of the business process needed to completely solve the business problem. In essence, they answer the question of "What?" not "How?".



Once the Business Requirements are defined, they can then be used as the basis for developing the Use Case diagrams. Specifically, each functional requirement identified in the Business Requirements will initially correspond to one Use Case if it can be automated. If it can't be automated, then a manual transition plan will need to be developed.

While it may appear redundant to develop a 1:1 correspondence between Business Requirements and Use Cases, it isn't because further refinement will be performed on the Use Case diagram during system analysis. The mapping is only 1:1 at the beginning of this stage. Once the Use Case diagram is refined with "Extends" and "Uses" relationships, the mapping becomes a many-to-many relationship.

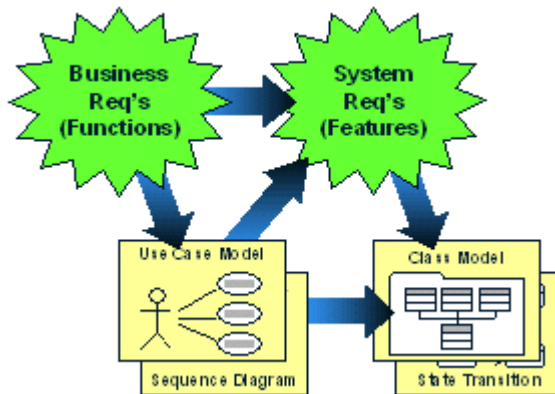
It is important, though, that this reuse analysis be performed on the Use Cases and not on the Business Requirements. This is to avoid corrupting the real business requirements with artificial technological constraints. All too often users are forced to redesign their business process in order to accommodate technology rather than the reverse. As the saying goes, "Just because you know how to use a hammer, not every problem is a nail". That is, define the business requirements for a solution before a specific technology is applied.



As part of the transition to system design, both the Business Requirements and the system analysis models are used to develop System Requirements. As with the Business Requirements, the System Requirements are stated in a textual format. However, unlike the Business Requirements, the System Requirements define the technical features of the application rather than the business needs. They are used to define

the "How?" for the application.

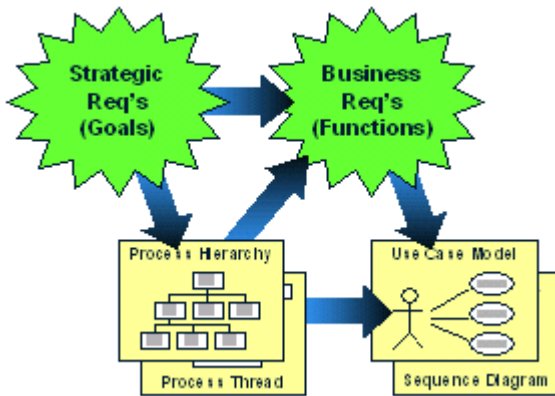
At this point, relationships should be established between the System Requirements and the previous deliverables. For example, each System Requirement should be an implementation of at least one Business Requirement and at least one Use Case. These relationships are then analyzed to look for inconsistencies in the model. For example, if any Business Requirement does not have at least one "child" System Requirement, there is a gap somewhere in the Use Case model. On the other hand, if there are System Requirements without at least one "parent" Business Requirement, then the scope of the original project has been increased, either intentionally or unintentionally.



Once the System Requirements have been fully defined and quality checked, they are used in conjunction with the Use Case model to develop the Class model. As with the System Requirements, relationships should be established to the "parent" objects in the previous deliverables and used to look for inconsistencies and omissions in the Class model.

Finally, the system design and implementation models are then used to develop the application code itself. By using this "matrix" approach to building the UML deliverables, the resulting application is more complete and of higher quality.

If a formal business process model is desired, the RBU process can be extended to support this work as well. Although the UML specification does not include notation for business process models, there are many popular methodologies which do. For example, the CSC Lynx method is used by many modeling tools to implement this work.



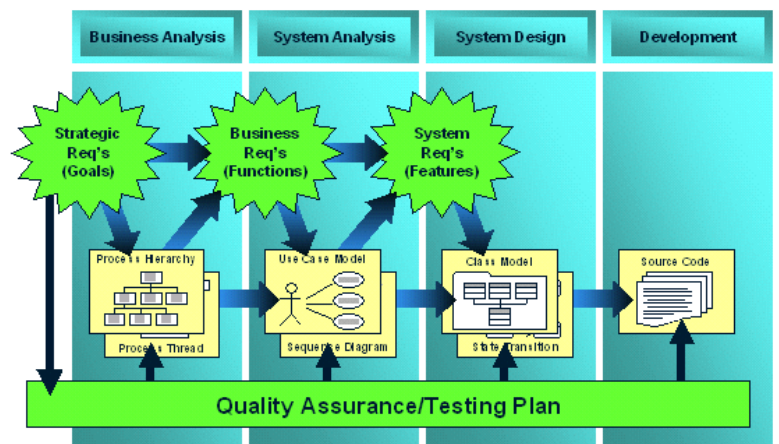
In these cases, additional diagrams are developed before the UML deliverables listed above are created. These might include models like a Process Hierarchy diagram and/or a Process Thread diagram. Both of these diagrams show the flow of work through an organization without regard for job titles and application boundaries. Their purpose is to optimize the organizational process before an application system is designed.

Using the RBU approach, the Process Hierarchy and Process Thread diagrams would be preceded by a set of requirement definitions. These requirements, called "Strategic Requirements" define the goals of the organization, including the objective metrics used to measure success. These Strategic Requirements become the guiding principles used to govern which possible business process is the most desirable.

As with previous requirements, Strategic Requirements are related to other requirements and the modeling objects. Specifically, Strategic Requirements should be related to the Business Requirements necessary to accomplish the goals and to the processes in the business models that implement them. Again, these relationships can be inspected for inconsistencies before moving forward in the development lifecycle.

Finally, as mentioned earlier, the RBU method includes a third parallel activity for Quality Assurance and Testing. This set of work is performed by the QA organization and is used to detect flaws in the requirements and models deliverables and to develop the test harness used for verification of the application code. By deriving the test cases from the requirements in a progressive manner, the resulting test plan will be more complete and should validate both functional and operational performance.

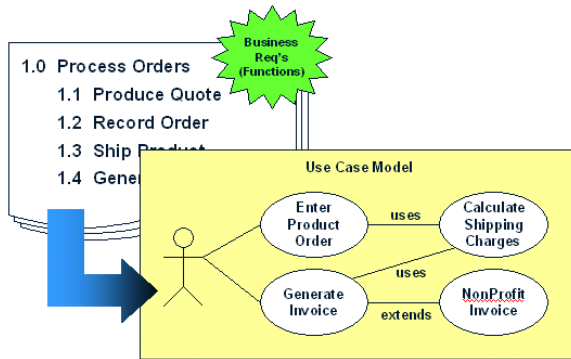
At first glance, the RBU approach may seem to introduce additional work on the project team because there are more steps than in a traditional UML-based method. However, in the opinion of this author, the work that RBU dictates is not additional work, but rather it is a matter of formalizing work that is already being done with informal methods. By purposefully addressing



these steps, the quality of the work will increase and productivity may actually improve. At the very least, the quality of the software product itself will be measurably higher.

RBU Example

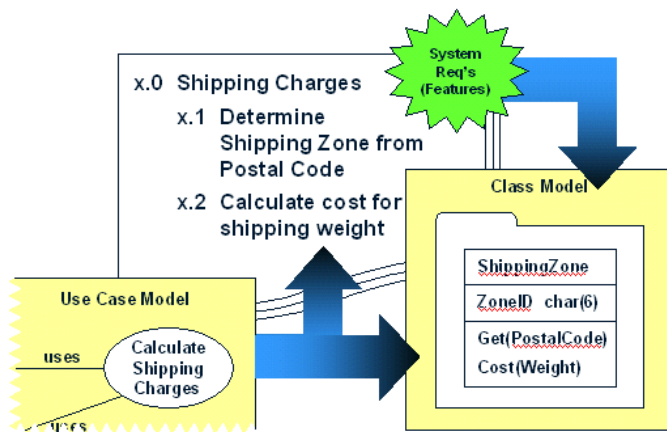
To demonstrate the RBU technique, consider the example of the "Enter Product Order" Use Case shown in the previous chapter. How did the user and/or development staff conclude that this Use Case was necessary? How did they know which system functions would be appropriate to solve the business problem?



Most often, the answer to these questions is that the Information Systems staff asked the customer what the solution should be. This assumes, however, that the customer has the information and expertise necessary to make this decision. Very often that is not the case and dangerous assumptions are introduced into the development

effort before a single line of code is written.

In an RBU-based project, the Use Case model would be preceded by a set of Business Requirements which document the business solution to the problem. In our example, the Business Requirements would include information about not only the two automated tasks ("Record Order" and "Generate Invoice") but also the two manual tasks that surround them ("Produce Quote" and "Ship Product"). Together, these four tasks identified in the Business Requirements describe the complete business solution necessary to solve the stated problem. Now the development team has sufficient information to make an informed decision about which tasks can be automated and how best to implement them.



Initially, only two Use Cases are defined. These would be named "Enter Product Order" and "Generate Invoice" and would be related to the two Business Requirements that are being automated. During the course of the system analysis, however, additional Use Cases might be identified in order to encapsulate reusable logic (e.g., "Calculate Shipping Charges") or to extend the model for alternate courses

(e.g., "Non-Profit Invoices"). These additional Use Cases would not be directly related to

Business Requirements but would instead derive their relationships through other Use Cases.

Once the Use Case model was completed, the System Requirements would then identify the functional and non-functional software features needed to automate the Use Case definitions. These System Requirements would then, in turn, be used to help define the objects in the system design models (e.g., Class diagram, State Transition diagram, etc.).

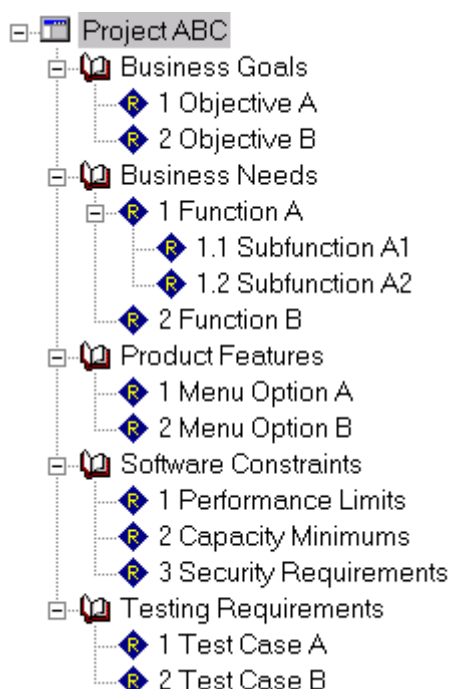
Benefits of the RBU Approach

There are many benefits to using the RBU approach instead of traditional UML-based methods that treat requirement as "features". The most important is that a structured requirements-based approach to development will dramatically improve the level of communication between end-users and the development staff. By providing a non-threatening textual format for deliverables, customers without training in UML notation are able to participate in the application specification. The requirements documents produced will be easier to read and more likely to be reviewed by the appropriate customers. All of this means more feedback which will lead to higher quality deliverables.

Another major benefit is that RBU provides a facility to document the entire business solution, not just the automated subset of it. Where UML modeling techniques make the assumption that a systemic solution is available, RBU requirements do not. This, in turn, provides a much richer picture of the solution and allows the project team to make more informed decisions about what automated functionality can and should be included in the application.

In fact, business improvements are often suggested as a result of the RBU requirements analysis that have nothing to do with application development. These solutions would typically not even be discussed during a UML-based project. Just as often, applications cannot efficiently solve the root cause of the business problem being solved because it is not an automated problem. It is important to understand this before an expensive development project is launched and then later cancelled due to lack of substantive results.

Finally, a third major benefit of RBU is that the scope of the application can be managed based on customer needs rather than on software features. By using the requirements as an integral part of the change control process, change requests can be evaluated on the basis of the business improvement in the Strategic Requirements and/or Business Requirements. Then, using the relationships established between the various RBU deliverables, the complete impact of a change can be determined before the change is approved.



Advanced Topics

Requirement Hierarchies

The requirements deliverables described in the preceding chapters are all intended to be document-oriented artifacts with textual statements of requirements. While this format is easier for end-users to review and approve, it can make it difficult to find specific sections when changing or searching the requirement information. To help resolve this problem, most

requirement deliverables are organized in requirement hierarchies.

A requirement hierarchy is simply a "tree-like" structure of requirements with similar requirements grouped into common "branches". The major branches of the tree correspond to the requirements deliverables listed above (Strategic Requirements, Business Requirements, etc.). The subordinate branches are defined based on the business area or organizational structure most appropriate to the project under development.

For example, in the Strategic Requirements analysis, the goals are most often subdivided by the organizational unit(s) being analyzed. Within each unit, then, the specific objectives might be identified and listed in priority order. However, during Business Requirements analysis, it may be more convenient to organize the sub-branches by business area or logical transaction.

No matter how the requirements hierarchy is organized, the most important aspect of the hierarchy is to understand and manage each requirement as an individual object within the set. Rather than treating a requirements document as a single block of text, each requirement should be treated as a separate entity and uniquely identified. In this manner, attribute identification, historical tracking, and object traceability can all be managed at the requirement level.

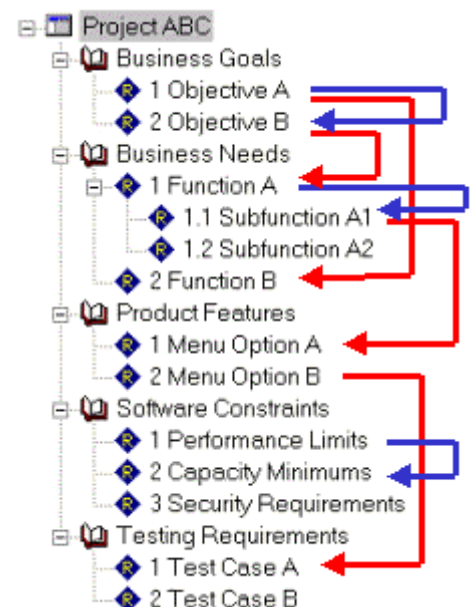
Requirement Traceability

Once the requirement hierarchy has been established, the rules for relationships between the requirements and the UML objects can be defined. These relationships, usually referred to as "traces", identify the dependencies between the various development objects. Typically, these traces are used to understand the impact of a request during the change control process as well as ensuring that changes are completely propagated throughout the development model.

For traces between requirements, relationships can either be established between two requirements on the same branch of the hierarchy or across branches. Most often, traces within one branch of the hierarchy are established to show a logical precedence between the two requirements.

For example, in the picture on the right there are two primary business goals that have been identified for *Project ABC*. These are named *Objective A* and *Objective B*. However, in addition to the textual definition of each, there is business rule that must be defined in the requirements document. Specifically, *Objective A* must be met before *Objective B* can be attempted. This precedence relationship is modeled as with traceability link between the two requirements, shown as a blue arrow in the graphic. It indicates that *Objective A* is the "logical parent" of *Objective B* and that any changes to *Objective A* must be reviewed to determine their impact on *Objective B* as well.

Traces between major branches in the requirements hierarchy, however, generally indicate a developmental dependency. That is, requirements in earlier stages of development should have traces to requirements in later stages of development in order to show the progression of the development effort.



In our example, *Objective A* is one of the business goals that must be met by the project. In order to show the development dependency to the business functions, a traceability link is established between *Objective A* and *Function B*. This indicates that *Function B* is necessary in order to accomplish *Objective A*. This relationship is shown on the graphic with a red arrow. Any changes to *Function B* must be reviewed to determine their effect on the project's ability to accomplish *Objective A*.

Generally, only "direct" traces are modeled between requirements. "Indirect" traces can then be implied by following the chain of the parent-child relationships. For example, in the *Project ABC* tree, *Objective B* has an indirect relationship to *Menu Option A* by following the chain through the intermediate nodes of *Function A* and *Subfunction A1*.

Traces are also established between the requirements and UML objects in much the same way. For example, if *Function A* was implemented by *Use Case 1*, a dependency trace would be defined between the two to show this relationship. Subsequent impact analysis could then be performed by following the trace from the requirement to the UML model or vice versa.

Without these traces, proposing changes during the development lifecycle becomes a subjective effort depending entirely on the memory of the requirements analyst(s). While this may occasionally be effective, most often it leads to understated estimates and inconsistencies in the software design. Requirements traceability makes change control an objective, rational process

References

[1] Grady Booch, Ivar Jacobsen, James Rumbaugh. *The Unified Modeling Language Reference Manual*. Addison Wesley Longman Inc., Reading, MA USA. 1999.

[2] Bernd Oesteriech. *Developing Software with UML*. Addison Wesley Longman Ltd., Harlow, England UK. 1997.

[3] Paul Allen, Stuart Frost. *Component-Based Development for Enterprise Systems: Applying the SELECT Perspective*. Cambridge University Press, Cambridge, UK. 1998.

[4] Pierre-Alain Muller. *Instant UML, English Translation*. Wrox Press Ltd., Olton, Birmingham, Canada. 1997.

[5] Martin Fowler, Kendall Scott. *UML Distilled: Applying the Standard Object Modeling Language*. Addison Wesley Longman Inc., Reading, MA USA. 1997.

[6] James Martin. *An Information Systems Manifesto*. Prentice-Hall International, London, England UK. 1984.

[7] Geri Schneider, Jason P. Winters. *Applying Use Cases: A Practical Guide*. Addison Wesley Longman Inc., Reading, MA USA. 1998.

ISCN Network based Quality Assurance

<http://www.iscn.com>

http://www.iscn.com/select_newspaper/qa-systems/nqa.html

Victory Project

<http://www.victory-group.net>

Profile

In September 1994 ISCN established a business firm in Dublin, Ireland, 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 sponsors and development partners. 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.

In August 1997 a development office was established in Villach and later in Graz, Austria to coordinate the integration of assessment and quality management solutions, co-developed by different partner consortia.

In Feb. 2001 an agreement was made to establish a training center in cooperation with the HUSEI (Hungarian Software Engineering Institute, as a central European SEI) in Budapest, offering a wide spectrum of quality management and assessment briefings, courses, and practitioner workshops.

ISCN's References

An Internet based teamwork platform solution for virtual offices and quality management in distributed teams has been developed in co-operation with the leading knowledge management provider Hyperwave . This system has been configured with ISO 9001 compliant processes and companies in different countries were ISO 9001 certified with such a system by different certification bodies. DNV experts even compared the solution with the one running in NASA.

ISCN has built a reputation of successful co-ordination of European Union projects over the last 8 years where we built software process improvement solutions, tools, knowledge libraries and joint dissemination and training initiatives.

In parallel to that we managed to establish an industrial partnership (non EU funded) of large Scandinavian, German and UK institutions to collaborate in an European Software Process

Improvement initiative.

Improvement teams have been working for leading automotive firms since 1994 with outstanding customers such as BOSCH, ZF, ZF Lenksysteme, Audi/VW, and a large set of middle sized firms in different countries.

Training courses and seminars have been held for and experiences were contributed from leading European industry such as: Delta, Denmark, IVF Centre for Software Engineering, Sweden, Southampton Institute, UK, SINTEF, Trondheim, Norway, University of Maribor, Slovenia, CISUC, Portugal, Siemens AG , Germany, Motorola Corporate Director , Software Quality Standards, Israel, Intecs Sistemi , Italy Industrielle Steuerungstechnik GmbH, Stuttgart, Germany, Valmet Automation Inc, Tampere, Finland, IDA Centre, Ireland, Joanneum Research, Graz, Austria, B-K Medical A/S, Denmark, Centraal Beheer, Apeldoorn, The Netherlands, Event AS, Oslo, Norway, INTRACOM, S.A Peania, Attica, Greece, Cortis Lentini s.p.a., Italy, Atos ODS S.A., Spain, Royal Institute of Technology, Sweden, Telenor, Norway, Brüel & Kjær, Denmark, TEGEA SA, Athens, Greece, ICL , Manchester, UK, Navia Aviation , Oslo Norway, Hüingsberg AG , Germany, Reis Robotics, Obernburg, Germany, ESBI Computing Ltd., Dublin, Ireland, Rational Software Scandinavia AB , Sweden, AEROPORTI di ROMA, Italy GMV, S.A., Spain, IVM Stuttgart, Germany, ISCN , Ireland, Hyperwave, Germany, NEC Corporation , Tokyo, Japan, HONDA Software Design Laboratories, Japan, Ericsson Telecomunicazioni SpA, Roma, Italy, School of Computer Applications, Dublin City University, Ireland, University of North London, UK, Computer Logic SA, Greece, CSELT, Torino, Italy, Defence Evaluation and Research Agency, Malvern, UK, Norwegian Computing Center (NR), Liebherr-Aerospace Lindenberg, Germany, PROVIDA ASA, Norway, SEKAS GmbH, München, Germany, Alcatel SSD , Antwerp, Centre de Recherche Public Henri Tudor, Luxembourg, Sztaki, Budapest, Hungary Kongsberg Ericsson Communications ANS, Norway, NASA IV&V Facility, USA, Nokia , Salo Finland, EDV Ges.m.b.H, Vienna, Austria, ELIOP, Madrid, SPAIN, Kapital IT, Denmark, ABS Group of Companies, Athens, Greece, Bosch Telecom, Frankfurt, Germany, DAKOSY GmbH, Germany, Sainsel, Sistemas Navales, S.A., Spain, etc.

ISCN's future mission is 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.

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 Union very well.

System Presented

Network based Quality Assurance Platform - NQA : Developed as an application on top of a tool box from the Hyperwave information server. Supports teamwork, workflow, task management, document management, version control. Is currently taken up in different projects at 2 million Euro for use in defense, aerospace, public service, and research networks.

Victory - An e-Commerce Platform with Setup Packages for Process Improvement

Developed in a joint action of 3 co-operating Leonardo projects - as an exploitation action. Investment of some 1,2 million Euro in total. Tool packages contain assessment tools, experience libraries, courses, and guidelines. Installs on Windows platforms with a menu system. Includes also a MySQL/PHP based e-commerce database system running on the net with info, ordering, and delivery services.

The **SINTEF** Group performs contract research and development for industry and the public sector in technological areas and in the natural and social sciences.

With 1929 employees and a turnover of NOK 1.4 billion, the SINTEF Group is Scandinavia's largest independent research organization. Contracts for industry and the public sector account for 90 percent of operating revenues.

IVF is one of Sweden's largest industrial Research and Training Organisations. Covering the entire product development chain IVF is involved in technology, process and management issues as well as working environment and human factors.

DSV Department of Computer and Systems Sciences, is a joint department between Stockholm University, SU, and Kungl Tekniska Högskolan (Royal Institute of Technology), KTH. DSV is one of the departments responsible for the new IT University in Kista Science Park. DSV joined EuroSPI in 2000 as a research partner in the fields of software process improvement.

DELTA is an independent centre for technology and innovation. DELTA develops new solutions, solves problems and transfers technology in: Electronics, software technology, light, optics, acoustics, vibrations and noise. Total staff is 220 and the annual turnover amounts to USD 23 Mill.

Defence Evaluation and Research Agency (DERA -> now **QinetIQ**). The UK's Defence Evaluation and Research Agency (DERA) is one of Europe's largest research organisations, employing 12,000 staff. DERA provides impartial services to its defence and civil customers in support of complex system procurement and development. DERA's System and Software Engineering Center (200 staff) are DERA's main focus in software development and best practice.

Arbeitskreis Software-Qualität Franken e.V. (**ASQF**) was founded in 1996. The ASQF e.V. is a group of software professionals whose aims are: to promote discussion and raise awareness of the important role software has to play in the wider community; to foster exchange of experience amongst software developers and quality managers; to underpin the sharing of knowledge between software developers from industry, research institutions and academia; and to encourage publication in the field of software quality.

American Society for Quality (ASQ) Software Division signed a cooperation agreement with EuroSPI to support each other in disseminating knowledge about SPI and working together on the creation of a body of knowledge of software engineering.

And **ISCN** (see previous chapter).

Internationally Organised by
European **S**oftware **P**rocess **I**mprovement
Partnership

www.eurospi.net



Locally Organised by
Limerick **I**nstitute of **T**echnology

www.lit.ie