SOFTWARE PROCESS IMPROVEMENT AT ABB – COMMON ISSUES AND LESSONS LEARNT

Christoph Welsch, Horst Lichter

In: Proc. of Software Quality Management SQM 97
Bath UK

SOFTWARE PROCESS IMPROVEMENT AT ABB - COMMON ISSUES AND LESSONS LEARNT

Christoph Welsch, Horst Lichter

ABB Corporate Research,
Heidelberg, Germany
{welsch, lichter}@decrc.abb.de

Abstract: The growing importance of software for products as well as processes has been recognised in several ABB companies. As a result they started initiatives to improve their software development. Unlike other improvement programmes in industry, ABB‘s software process improvement initiatives are not part of company-wide, globally controlled programme. They rather evolved locally in different ABB companies, coached and co-ordinated by ABB Corporate Research.

This paper summarises the experiences gained in the various process improvement activities. Firstly it describes how ABB‘s software process improvement initiatives relate to similar ones in other companies, and how the foci of the improvement measures evolved over time. Secondly, we present the current status of the improvement initiatives, pointing out the role of ESSI funded process improvement experiments in this context. In the main part we describe obstacles to software process improvements, and the lessons we have learned. We summarise our experience in terms of ten theses that we consider necessary conditions for successful software process improvement programmes. The theses cover technical, organisational as well as human aspects.

Keywords: Software Engineering, Productivity, Software Project Management, Process Models, ESSI, CMM

1 INTRODUCTION

Many companies in the software business have realised that substantial gains in productivity of software development and quality of software products can only be achieved by improving the software process. Such improvements have turned out to be costly and bearing a high risk of failure, since the software process is a complex system of relationships among processes, technologies and people.

Results and experiences of industrial software process improvement (SPI) initiatives have been recently published, such as Dion (1992) or Johnson (1994). These initiatives have typically been part of large, company-wide process improvement programmes. For instance Wohlwend and Rosenbaum (1994) describe Schlumberger’s software improvement programme.

The situation at ABB is different. ABB is a group of companies that operate in different countries and business areas. The companies are managed locally. As a consequence, smaller software process improvement initiatives have emerged independently at different ABB companies, with

specific improvement objectives. Consultants from ABB Corporate Research have been involved in many of these process improvements.

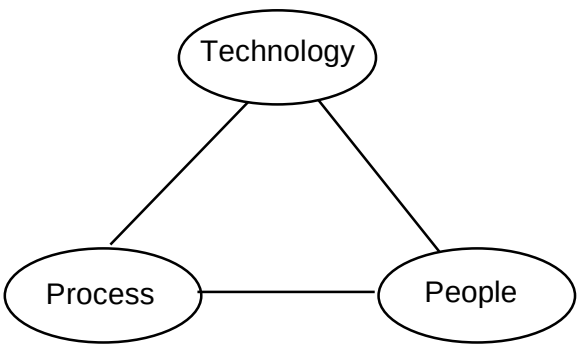
In this paper we present our experiences from software process improvement initiatives at ABB. We analyse common impediments and enablers of improvements, and summarise the lessons learned in terms of ten theses which we regard as important for successful software process improvement programmes.

The paper is organised as follows. Chapter 2 sketches the process maturity model underlying the improvement activities. Chapter 3 summarises the process improvement activities at ABB and describes three concrete instances in some more detail. In chapter 4 we present our experience and findings in terms of ten theses. Chapter 5 summarises our conclusions.

2 UNDERLYING PROCESS MATURITY MODEL

In general one can distinguish three key components that determine the productivity of software development and the overall quality of software products:

- The *process* specifies how software is produced. For instance it defines what stages or phases are used, what deliverables are produced, what the prerequisites are for moving from one phase to the next one, and what the responsibilities are.
- The *technology* comprises methods, languages, standards, and tools used in the development process.
- *People* eventually develop the software products, execute the processes and apply the technologies.



In order to systematically improve software development and its products all key components must be improved. During the last decade, the software engineering community has learned that the primary focus has to be on people and processes, then on technologies. Technology can only be efficiently used if people and processes have reached a certain level of capability.

The *Capability Maturity Model (CMM)* developed by the Software Engineering Institute at Carnegie Mellon University (Paulk et al. 1991) has strongly influenced the way process improvements are being conducted today.

The CMM is a well recognised standard which distinguishes five levels of software process maturity (see figure 2) and associates with each level a set of key practices which are required from software organisations on that level. The model also specifies how to advance to the next higher level by satisfying key requirements. Once a maturity level is known, the actions needed to move to the next level are more or less defined.

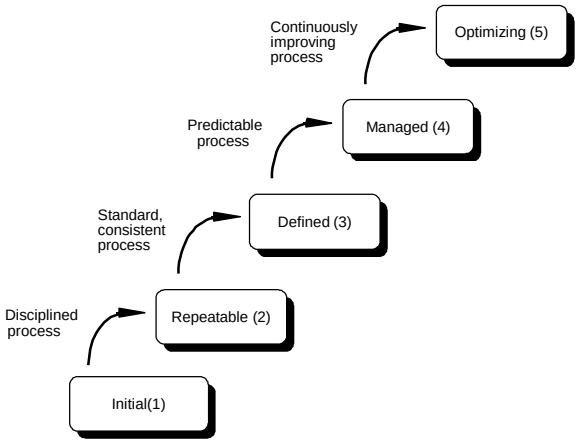


Figure 2: The CMM model

The CMM has become an industry standard for judging the capabilities of software development organisations. The effort for conducting process assessments has been published e.g. in Dion (1992). Herbsleb and Goldenstein (1996) present a systematic survey of CMM experience in industry.

The CMM includes an assessment technique to measure the software process maturity of an organisation. A CMM assessment is basically a series of questionnaire-based interviews. In order to apply the CMM in their organisation, some

companies have adapted the standard questionnaire to their needs (c.f. Anglade et al, 1993).

Early on ABB Corporate Research has adopted the CMM as a tool

- to make management aware of the necessity to improve software processes, and
- to guide and control the implementation of the improvements.

Most SPI initiatives at ABB have started with a CMM assessment. Some companies have also conducted follow-up assessments to measure the success of the improvements

3 ABB'S SOFTWARE IMPROVEMENT INITIATIVES

The major ABB locations dealing with software development are Germany, Sweden, Switzerland and USA. In all locations ABB companies have started initiatives during the last years aimed at improving their software development. Unlike other improvement programmes in industry, ABB's improvement initiatives have not been part of a company-wide, globally controlled programme. They rather evolved more or less independently, coached and co-ordinated by ABB Corporate Research.

The following chapters present three concrete instances of SPI initiatives from ABB Germany. Two of them (chapters 2.2 and 2.3) are funded as so called *Process Improvement Experiments* by the ESSI program of the European Community (ESSI stands for *European System and Software Initiative*).

3.1 SPI at ABB Kraftwerksleittechnik

ABB Kraftwerksleittechnik (KWL for short) is a company of the world-wide ABB group, employing 850 people, 90 of them in research and development (R&D). The improvement initiative reported here focused on the R&D department. Their business is developing control systems for power plants. The products range from controllers to operator stations and engineering systems. Software is a major part of these products.

3.1.1 Starting Scenario

Having recognised the impact of good software engineering on software costs, KWL started in 1991 an initiative to improve their software process maturity.

A team of people from KWL and the information technology department of ABB Corporate Research was in charge of defining and guiding software process improvement measures. They worked closely together with the operational units who eventually were to apply these measures.

3.1.2 Objectives

At the beginning, the improvements aimed at introducing new *software technologies*, such as CASE tools, or object-oriented programming, according to recommendations from a business analysis by external consultants. During implementation of the recommendations, however, it turned out that the software process at hand was not structured and documented well enough to allow efficient use of these technologies.

As a consequence the focus moved away from technologies to the *software process* as a whole.

After having defined a software process, the focus is now on improving testing activities of KWL's software products, as well as on configuration and change management.

3.1.3 Improvement Activities

Introducing a CASE tool. At the outset (1991) people believed that tools may help developers in solving their problems. After having selected and introduced a CASE tool it was very soon observed, that this was not true. It turned out that first a structured way of developing software has to be defined.

Definition of a process model. Next an activity was started to define the software process model. The findings of the first activity and a CMM assessment pointing out deficiencies in the fields of quality assurance, peer reviews, and configuration management provided the basis. The process model had to consider the organisational structures, the processes within and outside the R&D department, as well as major software engineering activities. The precise definition of tasks and responsibilities in the process model helped clarify organisational

boundaries and motivate organisational changes. Preparing templates and guidelines for documents defined by the process model was considerable work. Whenever possible we adapted published standards for this purpose, such as the various IEEE Software Engineering Standards (see IEEE 1994). This not only saved work, but also enhanced the acceptance by software engineers and management. The resulting process model was introduced in 1994 and is applied in development projects.

Introducing formal reviews. We learned that improvement measures are only effective when accompanied by verification processes. For this reason, the introduction of formal reviews was a cornerstone of all improvement activities. Formal reviews are characterised by a well defined review process and fixed roles of the participants; both aspects are essential for making reviews effective.

Improving testing efficiency. Although testing is a 'classical' activity, it is often done in a fairly casual and uncontrolled manner. At KWL, testing was improved by providing training in systematic testing, definition of test processes (module test, system test, system integration test) and their integration in the overall software process.

Change and configuration management. Currently there is an on-going activity that aims at improving change and configuration management. As a first result a change process together with a supporting tool was introduced in 1996. It is now used world-wide in the KWL organisation. The experiences so far are encouraging.

3.1.4 Results and Analysis

Looking back, it is striking that the most effective improvement measures were addressing management issues, not technology issues. The measures aimed at providing answers to "How-to"-questions: How to do and document project management? How to plan and implement configuration management? How to go about change management? How to assure software quality? How to manage testing?

It is customary in industry to bet on CASE tools when aiming for improvements in quality and productivity. We, and many others, have learned meanwhile that improving *processes* and *people* is far more effective than supplying new tools – yet it is also harder. For more detailed information about KWL's improvement activities we refer to Welsch (1995).

3.2 SPI at ABB Netzleittechnik

ABB Netzleittechnik (NET for short) offers to its customers a network control system, called S.P.I.D.E.R. which integrates functions like energy management, low and medium voltage distribution, supervisory control and data acquisition. The development of S.P.I.D.E.R. is distributed over three sites in three different countries.

3.2.1 Starting Scenario

In order to manage its software projects, NET has developed a software process model which takes into account all major aspects of software projects: organisation, planning, implementation, and control. Quality assurance is integrated in this process model as well. The process model is based on the traditional phased V-like model. It is applied in every development project. A software process group consistently maintains and improves the process model.

A particular weakness of the software process concerns the early detection of errors and the efficiency of testing. These findings were confirmed by a CMM assessment that was conducted in February 1995 by ABB Corporate Research.

3.2.2 Objectives

The overall objective of the improvement activities is to make validation steps more comprehensive and to reduce the effort needed. The specific objectives are:

- introducing formal peer reviews,
- defining and implementing a systematic test process,
- defining a tool environment supporting test activities, and
- measuring test activities and collecting relevant data.

3.2.3 Improvement Activities

Based on the results of the CMM assessment, an improvement action plan was worked out. Until now two major activities have been performed.

Introducing formal reviews. Although NET has a long tradition in reviewing documents, it was noticed that the results of reviews can be improved using a more formal peer review technique. In order to introduce this technique a training workshop has been organised.

Introduction of systematic testing methods. This activity is conducted in the context of the Systematic Module and User Interface Test (SMUIT) project, funded by the European Community as the Process Improvement Experiment 21612 of the European Systems and Software Initiative (ESSI). It is devided into the following phases.

PHASE 1: Defining the test process. The test process defines what activities have to be performed, what documentation has to be written, how these documents have to be validated and what the responsibilities are. Furthermore data about the testing process is collected during the experiment in order to have sufficient information for assessing the impact of the various test activities on software quality.

PHASE 2: Evaluation of test tools. Based on the experiences and results presented in similar evaluations available in the public domain, a small number of tools for detailed evaluation was selected. These tools were installed and evaluated. The tool selected has been integrated with the S.P.I.D.E.R. development environment.

PHASE 3: Performing tests. Before applying the new test practices the project members were trained both on systematic testing techniques and on the test process. In addition to general seminars on these topics, an introductory tutorial was given by the tool vendor in order to make the team members acquainted with its functionality.

After that, tests will be prepared, executed and documented according to the defined test process and supported by the tools.

PHASE 4: Validation of the experiment. Based on the collected quality data, the project team analyses the systematic testing approach introduced. The results obtained during the process improvement experiment will be summarised and disseminated by means of a final report. This report shall describe the experiment in a way that enables other companies to replicate the experiences.

3.2.4 Results and Analysis

The process improvement experiment SMUIT is going on. It is presently at the half way mark. The testing process that we have applied so far to analyse some S.P.I.D.E.R modules is suitable to deliver the following results:

- Identify those modules that are poor with respect to quality attributes such as complexity, readability, and maintainability.
- Identify those modules that contain dead, never used code.
- Identify those modules that are not testable due to their complexity with regard to the number of linear independent paths.
- Identify those modules that should be re-engineered.

Since we have analysed only a small part of the overall S.P.I.D.E.R. system so far, we are at the moment not able to provide quantitative results about the improvements achieved.

3.3 SPI at ABB Calor Emag Schaltanlagen and ABB Daimler-Benz Transportation

The two companies ABB Calor Emag Schaltanlagen (business: switch gear stations) and ABB Daimler-Benz Transportation (*Adtranz* for short; business: railway systems) faced similar problems in their control system engineering: The share of engineering costs in the overall product costs was steadily increasing. Poor tool support of engineering activities was identified as the major reason for this.

To improve the situation the two companies started a joint project developing a common integrated tool platform for control system configuration and maintenance. They decided to use object-oriented software technologies in this project (C++, bought-in class framework, object-oriented data base system).

Prototype development started in mid-1992 with a team of 5-7 members. Meanwhile the project has grown to about 30 people, distributed across three countries (Germany, Sweden, Switzerland).

3.3.1 Starting Scenario

The software process in this project had evolved in a fairly uncontrolled manner. With growing project size, the software process was found to be less and less adequate. The development project could be characterised by:

- *Ambitious project goals*. Development of an integrated engineering environment for control system engineering is a complex task. According to the classification in Goldberg (1995) it is a "first-of-its-kind" project for both involved companies.
- *Complex project structure*. The project is distributed across development teams in three countries.
- *No systematic analysis and design.* No published object-oriented analysis and design methods were used, nor any CASE tools.
- *Poor documentation of existing implementation.* The software underwent rapid changes with the effect that documentation became very quickly out of date.

When starting the improvement activities, the software process maturity at Adtranz as well as ABB Calor Emag Schaltanlagen was rather low. A CMM assessment which was conducted at Adtranz late 1995 and which also included the development project mentioned above, rated the maturity roughly at 2 on the CMM scale.

3.3.2 Objectives

As an answer to the realised deficiencies, upper management launched in January 1996 an initiative to improve the software process in the project. The objectives were:

- standardising forward engineering of object-oriented analysis and design,
- enhancing and automating documentation of work results, and
- reverse engineering of existing (insufficiently documented) object-oriented software.

Support for reverse engineering was not only needed for re-documenting the legacy software, but – even more importantly – for keeping the upcoming implementation (source code) and its documentation consistent. It is typical for most software projects that the source code evolves quite rapidly, while the documentation remains unchanged and is soon out of date.

To guide and coach the process improvements, software engineering consultants from ABB Corporate Research have been involved.

The improvement measures are funded as a *Process Improvement Experiment by the* European Community.

3.3.3 Improvement Activities

CASE tool evaluation. In the context of this process improvement experiment CASE tools were evaluated with emphasis on support for (a) forward engineering, i.e. analysis and design, and (b) reverse engineering, in particular for automated document generation.

The CASE tool that met the requirements best features a powerful documentation facility. It extracts comments and structures out of the source code and converts them, according to rules defined by the user, into text and graphics in a word processor template. So the source code is *the* single source both for code and design documentation. This *single source principle* increases the chance that in case of code changes also the pertinent comments – the source of the documentation – are updated.

Definition of a process model. The selected CASE tool supports the Coad/Yourdon method (Coad, 1991). For the overall process model, however, the method of Jacobson (1992) was felt better suited than Coad/Yourdon. So the management decided to use a mixed approach: the process and methods follow Jacobson, while Coad/Yourdon is used as the notation for object modelling.

The new software process prescribed well-defined documents as deliverables of important software engineering activities. As the first step, guidelines as well as templates for *Requirements Specifications* and *Design Descriptions* were provided. The requirements specifications follow

the pattern in figure 3, which adds chapters for object-oriented modelling to the IEEE standard structure for requirements specifications (see IEEE 1994).

3.3.4 Results and Analysis

To reduce the risk of the process improvement experiment we implemented the new practices in an incremental and iterative way. *Incremental* and *iterative* means that the improvements are being introduced by going several times in small cycles through the spiral depicted in figure 4. This approach gave us early indications of the usability of the new practices and of their acceptance by the engineers.

We regarded the acceptance by the project members as the key indicator for success. Only if the practitioners feel the benefits of the new software engineering practices pretty soon, they are willing to adopt them.

Standardised documents have turned out to be a good means to get order into a fairly uncontrolled software process. At the same time they allow to check how well the new practices are applied, and, if necessary, to take corrective actions.

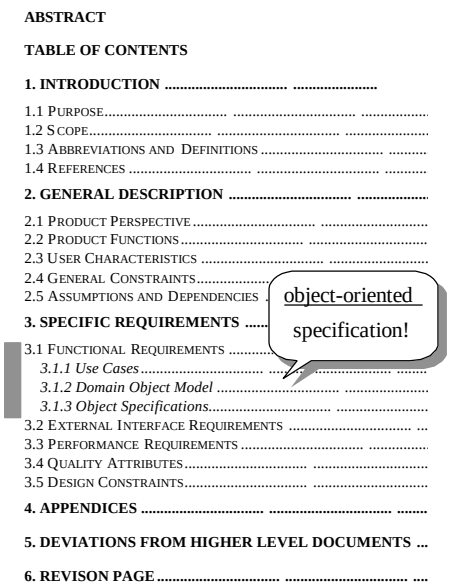


Figure 3. Template for requirements specifications

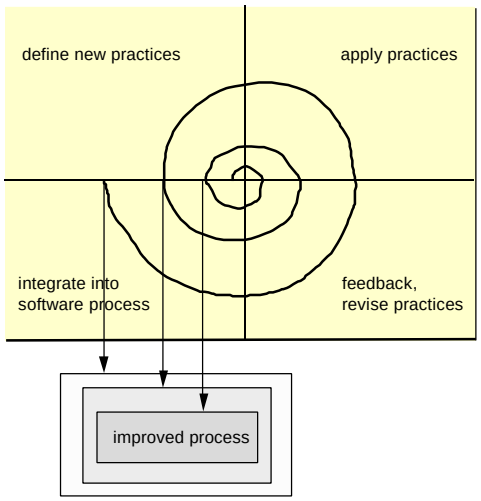


Figure 4: Spiral-like implementation of process improvements

4 LESSONS LEARNED: TEN THESES

In this chapter we would like to summarise our experience in terms of ten theses. They express findings that should be easily transferable to other organisations.

THESIS 1: Focus on people and processes, not on technology.

Process, people and technology are the key components of software development. Only if all components have satisfactory maturity the resulting products will be of satisfactory quality. Improvement of one component can not be achieved in isolation from the others. In the context of SPI that means particularly, not only to improve process elements (e.g. project planning), but also to enhance people's capability to perform the improved process. Only people can bring the improved process to life. This issue is the main focus of the *personal software process* documented in Humphrey (1995). Its objective is to make developers aware of the processes they use to do their work, and of the performance of those processes.

In summary we think that the emphasis of SPI activities should be on both the process and the people. As a consequence, we recommend not to introduce technology oriented tools before a software process is defined and applied.

THESIS 2: Basic improvements are management oriented.

We have observed that the majority of improvements addressed management-oriented issues. This conforms with the CMM which calls for good software engineering *practices* ("How to do it"), but does not prescribe certain *technologies* ("Which tool to use").

For instance, at the very beginning of KWL's software process improvement programme, we erroneously put our hope on introducing CASE tools. The CASE tool introduction failed (after considerable investments) mainly because the developers were not well prepared and the software process was neither defined nor applied well enough, to integrate the usage of CASE tools. The promised benefits of CASE can only be achieved if its use is part of a well-defined – and actually employed – software process.

A similar situation we encountered at the process improvement experiment at Adtranz and ABB Calor Emag Schaltanlagen. For engineers as well as management, the initial motivation to start a process improvement experiment was the wish to use some CASE tool. It turned out very soon, however, that management-related improvements (project planning and controlling) are more important in order to solve the main problems in the development project.

THESIS 3: No software process improvements without clear responsibility.

SPI is a long term activity. Significant results can only be achieved by continuously improving the development process. Therefore it is extremely helpful to have a group responsible for pushing SPI activities. This group, in the following called *software process group*, serves as a focal point where all SPI activities are planned, co-ordinated and assessed. This includes e.g. working out an improvement action plan or recruiting external consultants.

At KWL and NET the software process groups are also responsible for software quality assurance in general. At Adtranz and ABB Calor Emag Schaltanlagen, there was no such group. Here the management of the development project took over the responsibility for the process improvements and enforced them.

THESIS 4: No change of well-worn processes without new people.

Any group of people working together develops specific processes that have proven to work fairly reasonable. These processes are typically undocumented and communicated "by doing". The older the processes and the group structures are, the harder to change such processes.

Software process improvements aim at re-engineering existing processes. To make the re-engineering effective, positive examples are needed, that is, people that practically show how the new processes integrate with the daily work.

It is necessary therefore to bring new people with fresh ideas and views into the development projects. These people should also coach the project members in applying the new practices and processes. We have observed this especially at Adtranz and ABB Calor Emag Schaltanlagen where a group of new and highly motivated people was involved in the process improvement activities.

Involvement of external software engineering experts facilitates the acceptance of new measures. We consider it extremely difficult for an organisation to achieve substantial software engineering improvements by its own power alone.

At all ABB companies where SPI measures have been conducted (KWL, NET, Adtranz, ABB Calor Emag Schaltanlagen), people from ABB Corporate Research served as software engineering experts. At first (the initial SPI activities were started at KWL) the relation between the software process group, external experts, and development projects was as depicted in figure 5 (a): software process group and external experts devised in collaboration the improvement

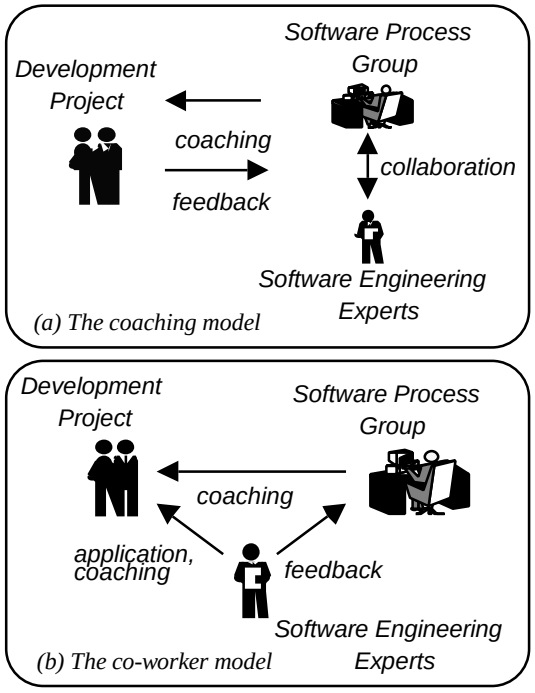


Figure 5. Organisational structures for implementing improvement measures (b is more effective).

measures, and coached the development projects.

It turned out that the structure in figure 5 (b) is more effective: Experts knowing the new practices co-work actively in the development projects. They not only introduce but also apply the new practices and feed back the application experience to the process group. That is, the software engineering experts participate actively in the software development (they are regular members of development teams), at the same time acting as links between operational and software process group.

THESIS 5: Process maturity assessments guide, control and sell the improvement process.

Software process assessments serve different purposes. We believe that, in the context of SPI, assessments are very helpful to support the following tasks:

- *Identifying the state of practice concerning the software process*. If you want to improve something, you have to know where you start from. The outcome of an assessment defines the baseline upon which improvements are to build.
- *Identifying the main deficiencies in the development process and planning the improvement activities*. Because a process assessment clearly identifies the main deficiencies, it supports planning the SPI activities, so that very important process areas (e.g. peer review) are improved first.
- *Quantifying the success (or failure) of improvement measures*. Generally it is quite difficult to quantify the benefits of SPI programmes (see also thesis 3.10). Although not expressing the return of investment in Dollars, the CMM is a widely accepted yardstick for measuring software process improvements. Even senior management understands it. Follow-up assessments are being conducted or planned at KWL and NET.

Furthermore, assessments and their results are a good means to build up awareness for SPI at the management level.

THESIS 6: Adoption of standards accelerates the implementation of improvements.

Many aspects of a SPI programme can be covered by published standard solutions. This is in particular true for several key practices of the CMM, which are addressed by IEEE Software Engineering Standards (see IEEE 1991). In the SPI programme at KWL, we have adopted the IEEE standards about project management, configuration management, quality management, requirements specifications and design descriptions.

Using standards has two big advantages over individually designed solutions:

- *Standards save work* because a lot of capable people put their knowledge and experience into the standards. You need not re-invent the wheel.

- *Standards receive far better acceptance* by management and developers than individually designed solutions. This is particularly true in an electrical engineering company like ABB. Here, most managers as well as software developers are electrical engineers by their education. They are used to standards and regard them highly.

THESIS 7: Pilot projects must be open minded towards innovations.

In the relevant literature one can find lots of recommendations on the characteristics of an ideal pilot project: it should be an important but not a vital project, be of decent size, but not too large, and the additional effort for the pilot applications should be planned and budgeted right from the beginning.

From our experience, an additional aspect is equally important to the success of pilot applications: The pilot project members should have an open attitude towards innovations. Only if they really want the new practices to be successful, one will have application success stories, necessary for dissemination of the new practices.

For instance, in the SPI initiative at KWL, we felt at times a somewhat "defensive" attitude towards innovations in pilot projects. Although not exactly rejecting the new practices, the pilot project members seemed to be not very interested in the success of these practices. In such cases the application of the new practices was more and more evaporating, and eventually substituted again by the former behaviours.

In the SPI at Adtranz and ABB Calor Emag Schaltanlagen, the situation was very different. Here, the success of the process improvement initiative was to the most part due to the enthusiasm and openness of the project members.

THESIS 8: The main obstacles to improvements come from organisational and human factors.

Any SPI initiative encounters difficult situations during its execution, sometimes even putting the whole initiative at risk. The causes for these difficulties trace mostly back to organisational and human factors. Some examples:

- The *lack of software engineering knowledge* in management and software development makes the management uncertain when it comes to assessing the value of new software engineering measures. Even the smallest crises may then endanger important parts of the SPI programme. At KWL, for instance, it took some time and considerable effort to convince the management of the necessity of a defined software process model.
- *SPI programmes are often felt as a threat*. They are mostly initiated by top management, aiming at improving productivity. Increased productivity, in other terms, means staff reduction. Therefore it comes as no surprise that many software developers get concerned, try to protect themselves against this threat, and are not very friendly towards the improvement measures (see also thesis 3.7).

THESIS 9: Software process improvements need long term management support.

SPIs are long term programmes of at least two or three years. They consist of a set of co-ordinated activities. Because these activities require a significant investment, the management of the development organisation must support the improvement programme actively. This means, that it is not sufficient to allocate the budget needed for the programme but also to demonstrate clearly that it is of very high importance. Experience shows that due to daily business needs improvement activities are often delayed or not conducted seriously. If management does not consistently back up the SPI programme, people's motivation and the intensity of the improvement activities are likely to decrease.

As figure 6 illustrates, there is a gap between the expected and the actual improvements. People expect more than is realistic. After some time, they realise this discrepancy and tend to be disillusioned. The initial enthusiasm has turned into serious doubts about the use of the SPI programme (the "valley of tears" in figure 6). This is a very critical phase. Only if management backs up the SPI programme, the benefits will be achieved.

Furthermore, the management has to ensure, for instance by audits, that the improved process is applied correctly in development projects.

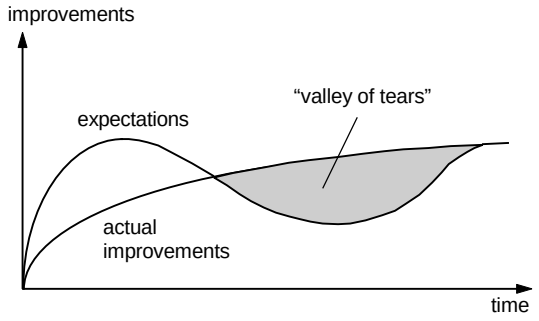


Figure 6. Expected versus actual improvements

THESIS 10: The investment in SPI is high, the return of investment is medium and long term.

Several results from SPI programmes in industry are reported in the literature. For instance, Hughes Aircraft started a two-year improvement programme to raise its Software Engineering Division from level 2 to level 3 (c.f. Humphrey et al. 1991). The programme cost the company roughly $400,000. They calculated that the initial return of investment (*ROI*) amounted to $2 million annually. Even more remarkable numbers are reported from Raytheon's programme: They invested almost $1 million annually in SPI, and achieved a 7.7:1 ROI, as well as 2:1 productivity gains (source: Saiedian and Kuzara, 1995). Larry Druffel from the Software Engineering Institute summarises in Methods & Tools (1994) that "*organisations engaged in process improvement for periods of three years or more achieved an increase in ROI of 4:1 to 8.8:1*."

For the SPI initiatives presented in this paper, the investment has been between 1 and 2 person years annually per SPI initiative. However, we are not yet able to express the ROI in quantitative terms.

5 SUMMARY AND CONCLUSIONS

Typical of all software process improvement initiatives at ABB – and we suppose this applies also to other companies – is that they start with emphasis on technologies, in particular on fascinating tools. As the initiative goes on, emphasis moves more and more away from tools towards processes and people's capabilities. This learning phase seems to be inescapable. We observed it in all improvement initiatives. The fascination of tools often serves as the "door opener" for more effective improvements regarding process and people.

The main risks for a successful and sustained software process improvement are over-ambitious improvement steps. It needs a systematic approach and a lot of experience to select the right set of measures suited for raising the maturity of the software process at hand.

We have made very good experience with the iterative and incremental approach to implementing software process improvements. It is more effective and bears less risk of failure than the "Big Bang" approach. The latter would first define the complete software process, and then try applying it.

A further danger for SPI initiatives is that their long-term character is not recognised. Process improvement involves change; changing established software processes takes time.

REFERENCES

Anglade, E., S. Miller, G. Tucker and A. Verducci, Jr. (1993) AT&T Software Process Assessments. *Knowledge Base*, Vol. 2, Issue 1, Jan.

Coad P., Yourdon E. *Object-Oriented Analysis*. Englewood Cliffs, Prentice-Hall, 1991.

Coad P., Yourdon E. *Object-Oriented Design*. Englewood Cliffs, Prentice-Hall, 1991.

Dion, R. (1992). Elements of a process-improvement program. IEEE Software, July, 83-85.

Goldberg A, Rubin K.S. *Succeeding with Objects — Decision Framework for Project Management*. Addison-Wesley, 1995.

IEEE (1994): *Software Engineering Standards Collection*.

Herbsleb, J.D., D.R. Goldenstein (1996): A Systematic Survey of CMM Experience and Results, Proc of ICSE 96, IEEE Computer Society Press, pp 323-330.

Humphrey W. S. (1995): *A Discipline for Software Engineering*, Addison-Wesley.

Humphrey W.S., R.T. Snyder, R.R. Willis (1991): Software process improvement at Hughes Aircraft, IEEE Software, July.

Jacobson, I. *Object-Oriented Software Engineering*. Addison-Wesley, 1992.

Methods & Tools (1994). 2(4), ISSN 1023-4918.

Paulk, M.C., B. Curtis, M. Chrissis (1991). Capability maturity model for software. *SEI Tech. Rep*. CMU/SEI-91-TR-24.

Saiedian, H. and R. Kuzara (1995). SEI Capability Maturity Model‘s Impact on Contractors. *IEEE Computer*, 28(1): 16–26.

Welsch, C. H. Lichter, M. Zeller (1995): Software Process Improvement at ABB Kraftwerkleit-technik, In Proc. of Experiences with the Management of Software Projects, Elsevier North-Holland (in preparation)

Wohlwend, H., S. Rosenbaum (1994). Schlumberger´s Software improvement Program. *IEEE Trans. on SE*, Vol. 20, No. 11, 833-839.

Johnson D.L., J.G. Brodman (1994): What small organizations say about the CMM,*" Proc. 16th Int'l Conf. Software Eng*.(ICSE 16), IEEE Computer Soc.