Process Improvement in Large-Scale Industrial Environments Based on SCM

Tilman Seifert and Markus Pizka
Technische Universität München
Institut für Informatik - I4
Boltzmannstr. 3
Germany – 85748 Garching

Karlheinz Raith
BMW AG
Sapporobogen 6-8
Germany – 80788 Munich

Abstract *Process improvement in a large-scale industrial environment is a challenge for numerous reasons. Besides others it is difficult to convince and motivate a critical mass of the staff and it is particularly expensive to train a large number of developers and managers. Established Software Configuration Management (SCM) practices pave the way to improve in a more efficient "bottom-up fashion" by processes based on SCM concepts. This paper describes the reasoning that justifies bottom-up process improvement based on SCM and presents our practical experiences with SCM-based process improvement in a large scale industrial environment.*

1 Introduction

Large-scale organizations developing software systems face a particularly difficult challenge when it comes to the definition and improvement of their development processes. For various obvious reasons, changes and improvement processes are difficult to enforce for large numbers of developers and even more difficult for large numbers of departments and subsidiaries.

Nevertheless, software development capabilities become more and more crucial for the success even in primarily non-software organizations. Several forces continuously demand further improvement: e.,g. high pressure on cost, time-to-system, and quality for critical systems. At the same time, new technologies introduce new tools requiring new methods and processes. One example for this is the transition from mainframe to multi-tier C/S-based and even web-based systems. Therefore process improvement is and will remain a key issue.

In large organizations there are several divisions and many developers in various projects with different needs for methods and tools. At the same time, it is particularly important for these organizations to use some common development model in order to be able to support knowledge sharing and learning at the level of the organization. Experiments with projects and tools can and should at first be made in individual projects, but the experiences should be transferred to other projects. This means, that decisions concerning new methods, tools, technologies, and processes have to be taken at the corporate level. They are of strategic importance for the whole company.

The German car manufacturer BMW has recognized the importance of software development for both, its products and its internal information management, several years ago. BMW participates in several initiatives to introduce and keep a high standard in internal and external software development and management. Due to the size of the organization, improvement steps must be introduced with great care in an incremental way. Here, we describe experiences made at BMW with gradual process improvement based on well-established techniques.

Goals

Improvement should be measurable. Therefore, the CMM [5] will be used as a metric for the process maturity. BMW's current aim is to achieve at least CMM level 3. This requires the introduction of a transparent and institutionalized process at the corporate level that is feasible for all divisions, generic enough to allow some tailoring for specific project situations, and concrete enough to give clear statements about transparency, defined roles, and communication.

Since software development at BMW is already done in a very disciplined manner, there is no need for a revolution — instead, an evolutionary improvement process should be installed that respects and takes advantage of the already achieved training

[1]Part of this work was sponsored by the German Federal Ministry for Education and Research (BMBF) as part of the project ViSEK (virtual software engineering competence center).

level, existing practices, and tools that are already in use. Furthermore, BMW's current improvement activities particularly emphasize the replacement of manual processes with tool-supported ones.

2 Initial Situation

This section briefly outlines BMW's software processes before the start of the improvement program described in this article.

2.1 BMW ITPM

The ITPM (IT Project Management) is a corporate guideline for software project management at BMW. It defines general rules about the way software projects are defined, initialized and carried out. It provides advice to project managers so that no important process area will be forgotten. However, the ITPM stays on a fairly abstract level of project description. It does not touch any concrete software engineering issues.

2.2 SCM Practices

BMW has extensive experience with Software Configuration Management (SCM) [7] mostly using *Continuus* [1]. SCM is used on all platforms from mainframe and Unix down to PCs. The introduction of practices and complex tools like elaborated configuration management tools is expensive but proved rewarding. Now that the knowledge about SCM and the tools is common within the organization, it is possible to do the next step in process improvement, taking advantage of the existing experience.

BMW uses an advanced understanding of SCM. The basic notion of SCM includes identification, control, status accounting, audit and Review. A broadened definition of SCM – usually dependent on the support of the tool being used – also includes process management and team work [3].

2.3 What's Missing?

ITPM does not define a complete software life cycle or give a detailed description of roles and tasks, and SCM practices used to be described in the manual of the SCM tool. Therefore BMW wanted a well-defined and complete software life cycle that integrates the SCM process as well as other already established and new processes.

This gap is bridged by the BWM-SLC which is described in section 4. The SLC combined with ITPM will define a complete process model for software development and project management within BMW.

3 Processes and Improvement

It is commonly accepted that product quality improvement can only be achieved by appropriate disciplined processes (see [4]). Among the numerous existing process improvement models, the most widely known and used are the Capability Maturity Model (CMM, see [5]) as well as several ISO standards, e. g. ISO 15504, better known as SPICE (Software Process Improvement and Capability dEtermination, see [6]).

3.1 Top-Down and Bottom-Up

Process improvement can be done top-down or bottom-up. In both cases, goals are defined that should be reached in the next improvement step. The interesting question is how these goals are identified. Top-down approaches postulate an ideal development process or requirements for an ideal process and derive the goals from there.

Bottom-up approaches on the other hand analyze the current situation of a development process and use the results to define the goals for the next step of process improvement. They consider the real situation and find goals that are reasonably reachable.

3.2 Improvement using CMM

CMM analyzes a number of key process areas in order to determine not only the quality of the process but in fact the capability of the organization. In this sense, CMM clearly is a top-down approach. It can be used for different purposes: When it is used for documenting the organization's capability, the results are ususally reduced to only the number of the CMM level that is achieved. But a CMM analysis has more interesting results to offer: An organization can use CMM for analyzing its own processes in order to improve the software processes; CMM gives a profile of the strengths and weaknesses of the organization's processes. Using this profile it is possible to identify the process areas to improve first.

Very important for improvement induced by CMM is the experience that improvement should be done step by step. Processes need to be lived by all affected groups and individuals. Therefore, CMM advises not to try and skip levels, and helps choosing the most important process areas where to start improvement. This is emphasized because a pure top-down approach would have the problem that the goals might not be set realistic.

One of the process areas that CMM suggests to introduce early in an improvement program is SCM. It is described as a process that supports the whole

life cycle of a project running continuously and parallel to other processes like the core activities (such as specification or testing), or other service processes (such as project planning and tracking).

Thus, CMM can be applied as follows: CMM indicates (from the top) that SCM is needed; then, in a second step, one has to determine how SCM can be introduced or improved. We followed this pattern by using CMM to analyze the BMW-SLC. The results of our analysis are summarized in section 5.1.

3.3 Other Improvement Models

Another common software process improvement model or method is SPICE. It chooses a similar approach like the CMM in the way it identifies certain process areas that need to be improved; it also uses five maturity levels and assigns the process areas to be fulfilled to the maturity levels.

Often referenced in in this context is ISO 9001. The ISO 9000 series was developed for any businesses, not particularly for software development; it concentrates mainly on customer satisfaction. ISO 9001 specifically addresses software organizations' issues and is widely used for process quality assessment.

A different improvement approach is taken by the Goal Question Metric model (GQM, see [2]) which is improvement in a bottom-up fashion.

3.4 BMW's Bottom-up Approach

Considering the lessons learned from proven process improvement models we briefly recall our goals: We need a practically applicable way of improvement in a large-scale organization in order to see improvement fairly soon. Improvement that is cost and time efficient motivates the developers and pays off in pecuniary terms.

If we want to realize quick improvements, we need to build on skills and experiences that the developers and project managers already have. That means we have to institutionalize processes that require practices that are already (at least partly) in use. Once processes are installed, they can gradually be improved and refined.

Experiences with process improvement are reported from both academic and industrial side, and they should be used for developing an improvement program for BMW. Both sides are best brought together by CMM (see section 3.2 and [5]).

Finally, we need a way to test whether improvement is successful, and to know in advance about the quality of the resulting process. CMM proved helpful for analyzing the planned new process.

Since we start process improvement by building on existing skills and practices, we call this approach "bottom-up". But the term bottom-up should not be confused or misread with a "lack of perspective": CMM helps not to lose overview, to keep the top in mind and to direct all improvement efforts to the most effective spots.

There are several process areas that need to be addressed, each dealing with software objects (or work products), (e. g. documentation or code), activities (how to transform one into another), and affected individuals or groups. Which one is the one to start with? There is one process that deals with objects as well as with activities and usually defines the role concept for all affected individuals and groups: the SCM process. As we will see, it is a solid basis for process description and for further improvement.

4 Multi-Tier Life Cycle

This section introduces the structure and the features of the BMW Software Life Cycle (SLC) for the development of multi-tier applications. Since SCM tools manage code as well as documents and tasks, and considering the fact that SCM is already widely used, it is a straightforward idea to describe the whole life cycle in the terms used by SCM.

4.1 Structure of the SLC

A software life cycle contains many processes. The SLC is a complete description of a software life cycle, containing all process areas that are commonly found in software engineering literature and in industrial practice. Some core processes are directly related to the software development itself like requirements management, specification, implementation, integration, testing, and delivery. Service processes like change management, configuration management, infrastructure management, inspection and test management are defined. Many of them run parallel and require continuous attention.

The SLC defines these processes separately and joins them, using the roles and tasks given by SCM. Thereby, configuration management advances from just versioning documents to controlling the workflow of the development cycle.

In the definition of the SLC the SCM process itself does not have a special place, its importance is not specially emphasized. But all other processes refer to the tasks and roles that are derived from the SCM process and the use of a SCM tool (*Continuus*, in this case). This leads to a straightforward and effective organizational integration.

4.2 Features of the SLC

The descriptions of all process aspects of the SLC are closely related to the tools that are used. This is true not only for the SCM process, but also e. g. for the requirements management, the infrastructure management, and the test management, to name just a few.

Wherever possible, the SLC descriptions refer to the project management description given by the ITPM (see section 2.1). This way the SLC gives a concrete realization of the project management definition of the ITPM. Here again, the SLC is the bridge between structuring the software engineering approach (bottom-up) and the definition of successful project management (top-down).

5 Current State

5.1 The SLC Analyzed Using CMM

We analyzed the documentation of SLC (and partly ITPM) using CMM. The aim was to find strengths and weaknesses *before* introducing the SLC broadly. The findings are interesting and encouraging. SLC (in combination with ITPM) lifts software development at BMW close to CMM level 3.

We contribute the qualities of SLC to the consequently followed rules of clearly assigning responsibilities, and defining, planning and documenting every step with the minimal overhead of using the SCM tool. This makes the SLC practical but not trivial.

Another success factor surely is the fact that SLC was built using currently available and "lived" SCM practices. This makes us confident that the introduction of SLC should not be technically difficult.

5.2 Experiences

Building a process definition on existing practices is efficient and proves to be successful. By this, widespread, well-documented and commonly accepted techniques are used to gradually advance practices to a defined process. The SCM process is a particularly good example for this. Defining a process on the basis of SCM makes it possible to introduce guidlines at the corporate level without the hazards of a revolution. Thus the bottom line of our experiences so far is: If you want to improve your process(es), start by understanding and improving your SCM process.

The success factor is to do both a top-down analysis of the aims in order to set the right goals, and a bottom-up improvement process to perform realistic steps based on existing capabilities.

6 Conclusion

In this paper we presented a concept for performing process improvement on the basis of existing and wide-spread techniques, tools, and skills by combining top-down perspective with bottom-up implementation. SCM was successfully used as the central starting point because SCM is perhaps the best known, broadest accepted and fully tool supported software development technology. At the same time, the conceptual foundation of modern SCM tools such as Continuus is rich yet flexible enough to allow the introduction of workflows and roles. By this process improvement can be performed as a gentle evolvement of existing processes and does not require radically new processes. This way, existing skills and experiences are reused and extended, and the major obstacle to process improvement, which is motivating a critical mass for the envisaged change, is circumvented. Our CMM case study with the BMW SLC demonstrated, that using an existing technology, such as SCM, as a basis for process improvement does not necessarily contradict with profound and long-term improvements perspectives. Here, the key is the combination of setting long-term goals by means of top-down analysis while enforcing actual changes by referring to existing and accepted tools and techniques.

References

[1] Continuus. www.continuus.com, 12 2002.

[2] Victor R. Basili. Software modeling and measurement: The goal question metric paradigm. Technical Report Computer Science, CS-TR-2956, University of Maryland, College Park, MD, 1992.

[3] A. Brown, S. Dart, P. Feiler, and K. Wallnau. The state of automated configuration management. Technical report, Software Engineering Institute, Carnegie Mellon University, September 1991.

[4] Mark J. Christensen and Richard H. Thayer. *The Project Manager's Guide to Software Engineering's Best Practices.* Best Practices Series. IEEE Computer Society, 2001.

[5] Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley, 1995.

[6] Spice - software process improvement and capability determination. Technical report, The SPICE Project, 1995.

[7] Bernhard Westfechtel and Reidar Conradi. Software architecture and software configuration management. In *Tenth International Workshop on Software Configuration Management*, 2001.