STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIII**, Number 1, 2008

SOFTWARE PROCESS IMPROVEMENT AT SYSGENIC

DUMITRU RĂDOIU AND MILITON FRENŢIU

Abstract. The Capability Maturity Model (CMM) was defined by Software Engineering Institute as a mean to improve the state of Software Engineering process. Going from CMM level i to the next level i+1 is seen as a major improvement. How such an improvement was obtained at Sysgenic is presented in this paper. Also, some consequences on the teaching process are presented.

Keywords: Software Process Improvement, CMM, education

1. Introduction

The term "software crisis" was introduced by the participants at two NATO conferences held in 1968 and 1970. It was observed that for a large number of software development projects deadlines are frequently missed, cost overruns are a rule not an exception, and it is increasingly difficult to measure the project progress. It was estimated [18] that more than 50% of the development project time is spent on testing and debugging. And the final product is not error-free; on average there still can be three to five errors for every hundred statements. We all expect the results given by our programs are correct, but 71% of software products have errors during their usage [16]. It is well known that some errors are not detected by testing, and some of them are never detected. Moreover, there are projects that have never been finished [4]. It is estimated that one from three large projects was never finished [7, 16].

There is a growing requirement of new programs. But our ability to build new programs hardly keeps pace with the demand for new programs. We have all observed the permanent and rapid growth of computer usage in all fields of human activity. The need for new programs is immense and their complexity is continuously growing. There are known today programs with millions of lines of code written by hundreds of people. These more complex programs cannot be developed like the old small ones. It has become necessary

Received by the editors: May 15, 2008.

2000 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.9 [**Software Engineering**]: Management – *Software process models*.

to analyze software costs over the entire life cycle of the system. It is known today that the major errors are due to errors in specifications, or poor design of the system, not to bad coding. After all, the fraction of time needed for programming is about 20% of the entire development process.

Among the factors that make software difficult four of them are inherently difficult [1, 14]. They are well known: the complexity of software, the conformity of the software product with the real world, the changeability and the invisibility of the product. The complexity of the problem affects the entire development process. It is difficult to understand the problem, to analyze it, to design and to implement the software product. And the real world cannot be changed to make the software product simpler.

The maintenance activity is a fundamental aspect of software engineering. It is a significant portion (exceeding 50%) of the total development process. But poor design and inadequacies resources threaten our ability to maintain existing programs.

To improve the state of programming activity the Software Engineering Institute approved a project to study this activity and to suggest ways of improving it. One of the project outcomes is a report [12,13] that contains the conclusions of that study and – based on the findings – suggests a number of steps considered to improve the software processes. According to this study the software companies are classified in 5 levels, defined by their performances. Key process areas specific to each level are also given.

Going from CMM level i to the next level i+1 is seen as a major improvement of the software processes of a company. In this respect CMM is seen as a guide for a continuous improvement.

Software Process Improvement (SPI) may be simply characterized by three main outcomes:

- respecting the cost and schedule of projects specified in the contracts;
- increasing the productivity;
- improving the quality of software products.

Reducing the rework is one possibility to increase software productivity, and to respect the schedule. We must build correct programs from the beginning [9, 11, 3]. As Gilb said: “Prevention is more effective than cure” [8]. Also, removing errors earlier permits to reduce the rework, and it is known today the importance of inspections (peer reviews) in this direction [6].

Building correct programs from the beginning is not a dream, today it becomes a reality. At IBM Mills introduced such a methodology, known as Cleanroom [10]. It has been successfully used for 20 years.

2. Reaching CMMi level 3 at Sysgenic

Sysgenic is a Romanian software development company with expertise in projects for financial and capital markets supplying customers in Europe and USA. The company software process improvement (SIP) started with documenting and institutionalizing ISO 9001:2000 requirements and this quality management system was certified in august 2005. The decision to implement a more professional quality management system (CMMi Level 3) was based on the need of more control over the projects based on more structured and practical project management principles. The outcomes were expected to be reflected in work performance, project visibility and control and in the end higher quality.

The Process Areas involved in CMMi Level 3 implementation are: Requirements Management (REQM), Project Planning (PP), Project Monitoring and Control (PMC), Measurements and Analysis (MA), Process and Product Quality Assurance (PPQA), Configuration Management (CM), Requirements Development (RD), Technical Solution (TS), Product Integration (PI), Verification (VER), Validation (VAL), Organizational Process Focus (OPF), Organizational Process Definitions (OPD), Organizational Training (OT), Integrated Project Management (IPM), Risk Management (RSKM), and Decision Analysis and Resolution (DAR).

First step was to set up the project team, also known as the Process Improvement Group (PIG), consisting of process oriented practitioners, with extensive experience in process design, software development and project management.

Second step was to provide them professional training (in CMMi) and documentation. PIG initial training started with an "Introduction to CMMI" SEI course, plus recent and extensive documentation on the capability maturity model integration (CMMi).

Third step consisted in initiating a "gap analysis" to document the differences (in the above mentioned areas) between what was implemented in the company and what are CMMi requirements. Based on these findings PIG initiated the design of the new internal process and started to implement them. Within the space of one year, these set of standard processes (OSSP) were institutionalized in the organization. Processes started to follow the new standards and to be documented accordingly.

After the processes were institutionalized and appeared to comply with the new requirements, the fourth step was an internal evaluation, also known as SCAMPI B. The differences between SCAMPI B findings and CMMi Level 3 required compliance were were smoothed up.

The following step is called "running in production mode": release internal process assets library, OSSP in organizational and software projects.

The on-site assessment, also called SCAMPI A, consists of

- pre-onsite period (consisting in collection and evaluation of project evidence), and
- on-site evaluation.

Following the on-site evaluation, Sysgenic achievement was recognized by Software Engineering Institute [15] in August 2007.

Now, one year later after being certified CMMi level 3, Sysgenic is following an internal QA audit on processes followed by a process improvement analysis and implementation on evaluation results. It is worth mentioning the constraints under which Sysgenic went into this SPI. Here are the most "visible" ones:

(1) making use of the already defined processes, known and largely used by employees;
(2) analyzing and deciding on the best usage of the existing tools (not always the best choice under CMMi level 3 exigency);
(3) a limited number of human resources who could be allocated to the SPI.

A first remark, following the above presented constraints, is that the newly CMMi Level 3 defined and institutionalized processes were sometimes time consuming and some even redundant. As projects are usually allocated small teams (8-10 people), any overhead generated by excessive documentation and/or training is "visible" in planning and cost and will lower the company competitiveness.

Following the first observation is that SPI never ends and the processes should be continuously reviewed and improved in successive iterations, focused mostly on:

(1) process and documentation simplification, maintaining compatibility with CMMi Level 3 and 4 requirements;
(2) identifying the most suitable tools which automate certain activities;
(3) processes institutionalization and periodic training (and re-training).

3. Conclusions

There are a significant number of lessons learned; here are some mistakes which we could have avoided.

- Make sure you have really experienced PIG team members, with a positive attitude, knowledgeable in their respective areas or expertise. During the process the PIG is usually overloaded, in a small company

they have to play multiple roles and therefore more likely to make errors. Their expertise and attitude, plus good planning as well as monitoring and control are essential in the success of the project.
- Simplify your processes to be more close to what we do, more natural to perform, to really help you in improving your organization performance.
- Develop your own simple and goal oriented metrics to document and track performance.

Software critical systems [2] require error-free programming and high quality software development processes. Obviously this requires also better educated work force, able to do this. It was a pleasant finding to learn that CMMi is taught at Petru Maior University (based in Tirgu Mures where Sysgenic HQ is located) as part of the Software Engineering course.

Attaining level 4 is, certainly, the next goal of Sysgenic. Quantitative process management and Software quality management are the Key Process Areas for this level. Continuous improvement, training and learning from our experience will help. A Software Metrics Program must be introduced to offer a quantitative feedback for improvement.

Nevertheless, we need more and more educated people as Software Engineers. And these people need a serious background from universities. Knowledge on Process Management, Verification and Validation (and consequences from the theory), Software Metrics must be present in their curricula [17].

There is a contradiction between the desire to obtain a system as quickly as possible, and to have a correct system. We need confidence in the quality of our software products. We need to educate the future software developers in the spirit of producing correct, reliable systems. For this we must teach students to develop correct programs. We are aware that usually programmers do not prove the correctness of their programs. There always must be a balance between cost and the importance of reliability of the programs. But even if the well educated people do not prove the correctness, their products are more reliable than the products of those "programmers" who never studied program correctness. Therefore, we consider that the students must listen, and pay attention to the correctness of their products.

It is unbelievable that students are superficially taught the theory of program correctness. As teachers, we must strive for a better education of the new generations of programmers. As scientists, we must look for better tools. The software development process must be based more on mathematical techniques, the formal methods must be taught and used as much as possible.

References

[1] Brooks, F.P., No Silver bullet: Essence and accidents of software engineering, IEEE Comput. 20, 4(1987), 10-19.
[2] Ricky W.Butler, Sally C.Johnson, Formal Methods for Life-Critical Software, NASA Langley Research Center, http://shmesh.larc.nasa.gov.
[3] Dromey, G., Program Derivation. The Development of Programs from Specifications, Addison Wesley, 1995.
[4] Effy Oz, When Professional Standards are LAX. The CONFIRM Failure and its lessons, Comm. A.C.M., 37(1994), 10, 29-36.
[5] M. Fagan, Design and Code Inspections to Reduce Errors in Program Development, IBM Systems Journal, 15 (3), 1976.
[6] Tom Gilb and Dorothy Graham, Software Inspection, Addison-Wesley, 1993.
[7] Gibs W.W., Software's Chronic Crisis, Scientific American, September, 1994.
[8] Gilb T., Software Inspection for the Internet Age: how to increase effect and radically reduce the cost, 2001, www.Result-Planning.com
[9] Gries, D., The Science of Programming, Springer Verlag, Berlin, 1985.
[10] Mills H., M.Dyer, and R. Linger, Cleanroom Software Engineering, IEEE Software, 4 (1987), 5, 19-25.
[11] Carol Morgan, Programming from Specifications, Springer, 1990.
[12] Paulk M.C., B.Curtis, M.B.Chrissis, C.V.Weber, The Capability Maturity Model for Software, Tech.Report, CMU/SEI-93-TR-25.
[13] Paulk M.C., B.Curtis, M.B.Chrissis, C.V.Weber, The Capability Maturity Model, Version 1.1, IEEE Software, 10(1993), 4, 18-27.
[14] Schach S.R., Software Engineering, IRWIN, Boston, 1990.
[15] http://sas.sei.cmu.edu/pars/pars_detail.aspx?a=9828 (retrieved 3^{rd} of June 2008)
[16] The Standish Group Report: Chaos, http://www.scs.carleton.ca/∼bean/PM/ Standish-Report.html
[17] ***. Computer Science Curricula at Babes-Bolyai University, www.cs.ubbcluj.ro
[18] Yourdon, E., Modern Software Analysis, Yourdon Press, Prentice Hall Buiding, New Jersey 07632, 1989

Petru Maior University, 1 Nicolae Iorga St., Târgu Mureş, Romania
E-mail address: Dumitru.Radoiu@Sysgenic.com

Babeş Bolyai University, 1 Mihail Kogălniceanu St., Cluj-Napoca, Romania
E-mail address: mfrentiu@cs.ubbcluj.ro